Z80 SOFTWARE DEVELOPMENT SYSTEM

Z80 SOFTWARE DEVELOPMENT SYSTEM

VERSION 2.1

April, 1980

COPYRIGHT

DISCLAIMER

Exidy Inc. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantibility or fitness for any particular purpose. Further, Exidy Inc. reserves the right to revise this publication and to make changes from time to time in the content herof without obligation of Exidy Inc. to notify any person of such revision or changes.

TRADEMARKS

EXLINK, EXASM and DEVCNVRT are trademarks of Exidy Inc. CP/M is a registered trademark of Digital Research.

TABLE OF CONTENTS

PART I EXASM

PART II EXLINK

PART III DEVCNVRT

PART I: EXASM

1. INTRODUCTION

EXASM (TM) and EXLINK (TM) together provide a means of transforming a Z80 assembly language program into an executable CP/M (TM) command program.

EXASM is a Z80 relocating assembler that recognizes standard Zilog Z80 mnemonics, and a useful set of pseudo-ops. It supports global symbols and assembles either relocatable or absolute modules. EXASM transforms the Z80 assembly language source program into an Intel hex format object (.OBJ) file, producing also, optionally a listing (print file).

EXLINK is a relocating linking loader which loads object files into specified memory locations and then optionally saves them as command programs onto CP/M disks.

It is not the intention of this manual to teach assembly language programming, rather to explain the use of the assembler and linking loader. Zilog's publication, Z80 Assembly Language Programming Manual, is an excellent reference, while the Osborne & Associates book, Z80 Assembly Language Programming, is good for learning the language.

NOTE

In all examples, underlines indicate operator input, and a carriage return is assumed at the end of each command line. As an example, here is how to transfer control from the A drive to the B drive. After getting the A> prompt, the operator types B: and a carriage return. CP/M responds with the B> prompt. In this case, typing B: (and the carriage return) is the only operator input.

```
A>B:
B>
```

2 DEFINITIONS

Some terms must be defined to use EXASM properly.

MODULE

A unit of code produced by an editor, loader or assembler. Another word for a program or program section.

SOURCE MODULE (usual file type = .ASM)

A source module is an ASCII text file composed of assembly language instructions--labels, op codes, mnemonics, operands, comments, etc. Source modules are created by editor programs, such as CP/M's ED, Exidy's EDIT or Exidy's Word Processor ROM PAC (TM). The assembler assembles the file into one object module. Lines are delimited by carriage return (0DH) or by carriage return/line feed (0DH, 0AH). EXASM supports the tab character (09H) and interprets it as a delimiter. The end of a source module is defined by the SUB or EOF character (1AH). The maximum source line length is the print line length minus 24; this is 108 characters as supplied by EXIDY (see 6.1.4 for print line length). The source module is machine code presented in a form readable by human beings.

OBJECT MODULE (usual file type = .OBJ)

This is a module produced by the assembler from the source module. Any object module contains machine code, (if relocatable, linking information), address and relocating information, and checksum information--all coded in ASCII. It is used by EXLINK. The format of the object module is an extended form of Intel hex format.

LOAD MODULE (usual file type = .COM)

A load module is a file consisting of the memory image of machine code for one complete program, created by EXLINK from one or more object modules and built in RAM. The file type is .COM, since it can be loaded and executed by using its name on the CP/M command line. Typing the name of a .COM file (without its file type) loads the program directly into the area beginning at 100H (CP/M's TPA--transient program area) and executes it. For this reason, EXLINK is used to relocate most object files to location 100H.

A symbol is an identifier of up to six characters (for more on this, see 4.3, Labels) which represents an address or constant. It may be defined by an EQU statement or by use in the label field of a source statement, or may be externally defined if declared in a GLOBAL statement. The assembler constructs the symbol table and the linker constructs the global symbol table. Symbols may be local or global; if global, they may be either external or internal.

LOCAL SYMBOL

A local symbol is one defined and referenced by one module only, and is not accessible to other modules. No record of any kind is made in the object module of a local symbol.

GLOBAL SYMBOL

A global symbol is one appearing in the operand field of linkage control type pseudo-ops. This set of pseudo-ops consists of GLOBAL, EXTERNAL, EXTERN, EXT, INTERNAL, INTERN, INT and PUBLIC. A global symbol is given global definition in a source module. Any global symbol in a source module appears in the corresponding object module. Once all object modules are loaded by EXLINK, all references to the global symbols of outside modules (or external symbols), are resolved, assuming there are no programmer errors in global symbol use. A global symbol is defined in one module and that definition is made available to other modules; the linker subsequently supplies the needed reference points.

INTERNAL GLOBAL SYMBOL

A symbol declared global (by the GLOBAL, INTERNAL, INTERN, INT or PUBLIC pseudo-ops) whose definition is found within the module is said to be an internal global symbol for that module. Its value is made known to all other modules loaded with it by EXLINK. When an object module is loaded by EXLINK, the internal symbol value is placed in EXLINK's global symbol table. These values must be addresses, not constants. That is, internal symbols are always relocated. The internal symbol has a value relative to the start of the module assigned by the assembler. EXLINK relocates this to an abolute address by adding the base address of the module within the final linked load module.

EXTERNAL GLOBAL SYMBOL

A symbol, declared global (by the GLOBAL, EXTERNAL, EXTERN or EXT pseudo-ops), which is not defined within a module, is an external global symbol with respect to that module. When this object module is linked with the

module where the symbol is an internal global symbol (that is, defined and declared global, the reference to the symbol is resolved. An external global symbol may never appear in an expression with operators or as the operand of an EQU pseudo-op in a source line.

POSITION INDEPENDENT

A program written so it may be placed anywhere in memory and still run properly without change is said to be position independent. Relocating information is not needed in the object module.

ABSOLUTE

An absolute program is one written without relocating information in the object module. A program is declared absolute by using the assembler pseudo-op PSECT ABS. An absolute program may or may not be position independent. Usually such a program can reside only in one area of RAM.

RELOCATABLE

A relocatable program is one without a PSECT pseudo-op, or one that has been declared relocatable with the PSECT REL statement. The assembled object file contains the object data which requires relocation if the intended execution base address is not the ORG value. Object address references are stored as values relative to the ORG value, as shown in the assembler listing. A relocatable program is usually position dependent, though not necessarily.

LINKABLE

An object module containing data about internal and external global symbols is a linkable object module. The loader uses this data to supply the absolute addresses in order to connect external references to internal symbols in modules. A linkable program may be either absolute or relocatable and may or may not be position independent.

TWO PASS ASSEMBLER

EXASM is a two pass assembler, that is, an assembler that scans twice each source module it assembles. Each scan is called a pass. During the first pass values for each symbol are determined and placed in a symbol table.

During the second pass, the assembler uses the symbol table created during the first pass to decode operand expressions into machine code. While assembling each line of source code, the assembler counts with its program counter each byte of object code produced. If no starting value is assigned by the ORG pseudo-op, then EXASM assigns a starting value of zero. The assembler also optionally suppresses creation of the object module and optionally produces a listing, with or without cross references specified, during the second pass. A linked list within the object data is created in the second pass for each external global symbol reference in the module, and a dictionary of global symbols is written to the object file. Diagnostic error messages are produced at all times in the second pass.

EXASM recognizes the standard Zilog Z80 mnemonics and a number of pseudo-ops (assembler directives). Assembly source modules are usually stored on disk under the file type .ASM. You may use EXASM to assemble files having other file types, so long as these have the same format as an .ASM file. (You could, for example, assemble directly a file created by Exidy's Word Processor ROM PAC, without the necessity of changing its file type from .WPF to .ASM.) EXASM assembles any file having the proper format (written, that is, in Z80 assembly language) into an object file. It also optionally suppresses or outputs an assembly listing (a print file) on the user's printer or on a CP/M disk. The object output of the assembler is a file in ASCII hexidecimal format with file type (if not otherwise specified) .OBJ, as:

```
FILENAME.OBJ
```

where FILENAME is the same name as that of the .ASM file. (The name of the file may optionally be specified to be different from that of the source file, as you will see later in the examples.)

Z80 source code input to EXASM is assumed to be a CP/M disk file generated by a CP/M text editor or by Exidy's Word Processor ROM PAC and disk interface.

3 OPERATION

The EXASM programs are used on a single or dual disk drive, such as Exidy's Display Disk System or Floppy Disk Subsystem. After connecting the disk drive, do the following:

Turn on the Sorcerer and all peripheral devices, including the disk drive unit.

Boot the CP/M system diskette. See the Exidy publications EXIDY CP/M and Display Disk Unit Operation Manual or Floppy Disk Subsystem Operation Manual.

Insert a CP/M system diskette containing the EXASM program and the file to be assembled into the disk drive and type the command listed in the next section for both EXASM and the file to be assembled existing on the same disk. EXASM and the file to be assembled may be on different disks. Examples are given in the next section. If EXASM is on the B drive and you wish to assemble your file on the B drive, after logging onto the A drive, by typing:

```
A>B.
B>
```

3.1 EXASM Call Format

```
A>EXASM <sourcefile>[,<objectfile>][,<printfile>] [/<options>]
```

<sourcefile>, <objectfile> and <printfile> may each have these properties: The name of the file may be any valid file name, up to eight characters, plus optional file type of up to three characters (and separated from the file name by a period). If no file type is specified, the type defaults to .ASM for the source, .OBJ for the object, and .PRN for the print file. If no file names are specified for <objectfile> and <printfile>, their names default to the same as <sourcefile>. The file name may be preceded by a drive identifier (any valid CP/M drive, such as A: or B:). If no drive is specified, EXASM defaults to the drive currently logged on. <sourcefile> is assumed to be located on the currently logged drive, unless otherwise specified (by preceding the name with a valid CP/M drive). If you specify only the drive name for <objectfile> or <printfile>, the output is directed to the specified drive. (Do this only if output is to go to other than the drive currently logged on.)

Delimiters should be used as shown. That is EXASM must be followed by at least one space, and, if <objectfile> and <printfile> are specified, items should be separated by commas. If <printfile> is specified but <objectfile> is not, two commas must be placed after <sourcefile>.

Options are specified by the use of a slash (/) followed by a string of characters consisting of one or more options. Options are not separated by delimiters. Options are explained in the next section.

In each of the following examples, in response to the prompt of the currently logged disk you type a command line following the prompt. So, if the A drive is currently "up" and you wish to assemble MYFIL.ASM:

Example:

Before assembly, this is what you have on the disk in the A drive:

```
A> DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM
```

Now, use EXASM.

```
A>EXASM MYFIL
```

EXASM searches the directory of the disk in the A drive for a file named MYFIL.ASM and if it does not find the file, it outputs the message SRC INPUT FILE NOT FOUND. EXASM is interested only in MYFIL.ASM and pays attention to no other MYFILs (MYFIL.HEX or MYFIL.COM, for instance, or even MYFIL with no file type). Before beginning assembly, EXASM always signs on, thus:

```
EXIDY Z80 Assembler - version x.x
Copyright (C) 19xx by EXIDY INC
```

If MYFIL.ASM exists and is an error-free Z80 program, this is what you see on the screen after the assembly is complete and following the sign-on message:

```
PASS 2

ERRORS=0000

WARNINGS=0000
```

We will give examples later of error and warning conditions.

Look at the directory (with the CP/M DIR command), and you find that two new files have been produced on the A disk:

```
A> DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM
A: MYFIL    OBJ
A: MYFIL    PRN
```

Note that if MYFIL.OBJ or MYFIL.PRN exist before the EXASM run, EXASM deletes them and creates new versions.

Example:

A> EXASM MYFIL.ASM

If MYFIL.ASM exists on the A drive, then the results are precisely the same as those of the previous example.

Examples:

A> EXASM B:MYFIL

or

A> EXASM B:MYFIL.ASM

If MYFIL exists on the B drive, then it is assembled as before, producing these two new files on the A drive:

```
A: MYFIL    OBJ
A: MYFIL    PRN
```

Examples:

B>EXASM MYFIL

or

B>EXASM MYFIL.ASM

EXASM searches the directory of the B drive for MYFIL.ASM and, if it exists, assembles it, producing on the B drive the object and print files.

```
B>DIR
B: EXCOPY   COM
B: EXASM    COM
B: EXLINK   COM
B: MYFIL    ASM
B: MYFIL    OBJ
B: MYFIL    PRN
```

Example:

```
A>EXASM MYFIL.WPF
```

EXASM looks in the directory of the A drive for a file named MYFIL.WPF and, if it exists, assembles it, producing these object and print files:

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    WPF
A: MYFIL    OBJ
A: MYFIL    PRN
```

Example:

```
A> EXASM MYFIL,PROG1,PROG2
```

Assuming that MYFIL.ASM exists on the A drive, EXASM assembles it, producing object and print files with the specified names, and the default types.

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM
A: PROG1    OBJ
A: PROG2    PRN
```

Example:

```
A> EXASM MYFIL.WPF,PROG1.XXX,PROG2.YYY
```

Assuming that MYFIL.WPF exists on the A drive, EXASM assembles it, producing object and print files with the specified names and types.

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    WPF
A: PROG1    XXX
A: PROG2    YYY
```

Example:

```
A>EXASM B:MYFIL,PROG1
```

EXASM assembles MYFIL.ASM from the B drive, producing two new files on the A drive (and leaving the B drive unchanged):

```
A>DIR B:
B: EXCOPY   COM
B: EXASM    COM
B: EXLINK   COM
B: MYFIL    ASM

A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: PROG1    OBJ
A: MYFIL    PRN
```

Example:

```
A>EXASM B:MYFIL,,PROG2
```

EXASM assembles MYFIL.ASM from the B drive, producing:

```
A>DIR B:
B: EXCOPY   COM
B: EXASM    COM
B: EXLINK   COM
B: MYFIL    ASM

A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    OBJ
A: PROG2    PRN
```

The two commas indicate no file name specified for <objectfile>, so the name defaults to that of <sourcefile>, with type .OBJ and assembled to the logged on disk, A. A new file name is given for <printfile>, so it is created as specified, with the default to the .PRN file type.

Example:

```
A> EXASM MYFIL,.XXX,.YYY
```

EXASM assembles MYFIL.ASM, producing object and print files with the default names and specified types.

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM
A: MYFIL    XXX
A: MYFIL    YYY
```

Example:

You may wish to assemble a file from one drive and send the <objectfile> and <printfile> to another drive. Before beginning EXASM, this is what you have on each disk:

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM

A>DIR B
B: EXCOPY   COM
B: EXASM    COM
B: EXLINK   COM

A>EXASM MYFIL,B:,B:
```

After the EXASM operation, this is what you have:

```
A>DIR
A: EXCOPY   COM
A: EXASM    COM
A: EXLINK   COM
A: MYFIL    ASM

A>DIR B:
B: EXCOPY   COM
B: EXASM    COM
B: EXLINK   COM
B: MYFIL    OBJ
B: MYFIL    PRN
```

3.2 Command options

Options are specified with a slash (/) followed by a list of single characters, according to the following table. Individual options are not separated by delimiters. If they conflict, the last-named option takes precedence.

C - Generate cross-reference (explained later).

D - Listing to disk. This is a default option and need not be specified unless you have turned off the listing option (bit 0) in location 103 (see Customizing EXASM).

E - "Ecology" or compressed listing--paper-saving option. Page ejects are not performed. Thus, the pseudo-op EJECT is ignored, as is the form-feed normally associated with TITLE and that generated at the end of the listing.

F - Set form-feed option. This option is used for printers which can handle an ASCII form-feed character. Since this is a default, you need not use it unless you have turned off this default in location 105 (see Customizing EXASM). The opposite of this option is option S.

G - Suppress generated text (beyond four bytes) of DEFB/M/W. You can make this a default option (see Customizing EXASM, where a full explanation of this function is given).

K - No listing or cross reference. Suppression of print file.

L - Listing to list device. Print file goes to the CP/M LST: device, usually a Sorcerer Centronics-compatible parallel printer device.

N - No object output. Does not produce an object file. This has no effect on listing or cross reference. This is useful for a fast syntax check of source.

O - Object output (default). Use if bit 2 of location 103 has been turned off (see Customizing EXASM).

S - No form feeds. Uses carriage return/line feeds instead. For printers that do not support form feed. This can be made a default (see Customizing EXASM), but is not supplied that way in the distributed version.

T - List to console, CP/M CON: device.

W - Don't print warnings. This too can be made a default.

Note that diagnostic messages always go to the console if listing is suppressed.

Examples:

```
d>EXASM SOURCE
```

produces

```
Source = d:SOURCE.ASM
Object = d:SOURCE.OBJ
Listing= d:SOURCE.PRN
```

where d=any valid CP/M drive.

Default options:

Listing will have form feeds. Generated text and warnings will be printed. Object and listing (with no cross reference) go to disk.

```
d>EXASM SOURCE,OBJECT,PRINT/SW
```

produces

```
Source = d:SOURCE.ASM
Object = d:OBJECT.OBJ
Listing= d:PRINT.PRN
```

Options:

```
S - Listing will have no form feeds.
W - Warnings will not be printed.
```

```
d>EXASM B:SOURCE.BAK,,PRINT.ECH/N
```

produces

```
Source = B:SOURCE.BAK
Object = no object generated (N option)
Listing= B:PRINT.ECH
```

```
d>EXASM SOURCE,B:/LC
```

produces

```
Source = d:SOURCE.ASM
Object = B:SOURCE.OBJ
Listing= to list device
```

Cross references will be included in listing.

For a quick syntax-check assembly use the options /NK to produce no object or listing, but just output diagnostic messages to the console.

3.2.1 Cross reference only assembly. The option string /LKC causes the cross references to list on the printer. /KC causes the cross references (only) to write to the specified disk file.

3.3 Interrupts

The operator may abort assembly with CONTROL C. This produces the message:

****ABORT ERROR = Z, OPR REQUESTED ABORT **********

CONTROL S stops assembly (during Pass 1 or Pass 2) until another character is struck. Use this with the T option to pause console output, which otherwise prints too fast to read, or, perhaps, to momentarily pause printer listing. (Be sure to hold the key down until it is acknowledged.)

4 SYNTAX

An assembly language program or a source module is made up of a sequence of source lines comprised of delimiters, labels, op codes, pseudo-ops, operands and comments in a sequence which defines the user's program. There follows a discussion of the syntax of the EXASM assembly language.

4.1 Source code format

The source code format requires the use of delimiters, to separate labels, op codes, pseudo-ops and operands from each other. The source code line format is:

[label] <op code> [<operand>] [,operand] [;comment]

Where expressions within square brackets ([]) are optional, while those within angle brackets (<>) must be supplied according to the conventions of Z80 assembly language programming.

4.2 Delimiters

Delimiters are one or more ASCII commas or spaces used to separate labels, op codes, operands, and pseudo-ops from each other. Carriage returns and semicolons are terminal delimiters, that is, they terminate the source line to be parsed by the assembler.

4.3 Labels

One or more characters compose a label. However, the assembler recognizes only the first six characters of a label. Control characters and the following ASCII characters cannot be used in a label:

' ( ) * + , - < > = . / : ;

Also, the first character of a label cannot be a decimal number. All labels must begin in column 0 unless followed by a colon (:). A label may be used on any line in the source module (with the exception of ENDIF). The value assigned to the label, if it is not before an EQU pseudo-op, is that of the current program counter.

4.4 Expressions

4.4.1 Constants. Constants must be in the range 0 through 0FFFFH. They may take these forms:

DECIMAL	- (default); any number with no qualifier is assumed by the assembler to be decimal. Numbers may optionally be qualified with a D. Examples: 34, 183D.
HEXADECIMAL	- must begin with a number (0-9) and end with H. If the first digit is a letter (A-F), a leading zero is added. Examples: 20H, 0A1DH, 0FFFAH.
OCTAL	- must end with Q or O. Examples: 327Q, 177O.
BINARY	- must end with B. Example: 01101010B.
ASCII	- must be enclosed in single quotes. EXASM converts them to ASCII hex code. Example: 'A' (= 41H).

4.4.2 Expressions. EXASM accepts many expressions in the operand field of a statement. Expressions are evaluated from left to right according to this hierarchy. (0 is the highest in this hierarchy, that is, it has the tightest binding power.)

operation	operator	hierarchy
equal to	= or .EQ.	0
signed less than	<	0
signed greater than	>	0
signed less than or equal to	<= or =<	0
signed greater than or equal to	>= or =>	0
not equal	>< or <> or .NE.	0
unsigned less than	.LT.	0
unsigned greater than	.GT.	0
unsigned less than or equal to	.LE.	0
usigned greater than or equal to	.GE.	0
reset overflow	.RES.	0
unary plus	+	1
unary minus (two's complement)	-	1
logical NOT (one's complement)	.NOT.	1
multiplication	*	2
division	/	2
addition	+	3
subtraction	-	3
logical AND	.AND.	4
logical OR	.OR.	4
logical XOR	.XOR.	4
logical shift right	.SHR.	4
logical shift left	.SHL.	4
modulus function (remainder)	.MOD.	4

(Expressions within parentheses are evaluated first, so you may use parentheses to change the order of expression evaluation.)

Examples:

In this expression:

```
3+2*4
```

first 2*4 is evaluated, then it is added to 3. If you wish to change the order, so that first 3 is added to 2, and then the result multiplied by 4, use parentheses, thus:

```
(3+2)*4
```

In this expression:

```
.NOT.X.AND.Y
```

First the expression .NOT. X is evaluated, and then that is ANDed with Y. That is, the expression is evaluated as if it were written:

```
(.NOT.X).AND.Y
```

In this expression:

```
A=B.OR.C=D
```

First the expressions A=B and C=D are evaluated, and then the result of A=B is ORed with the result of C=D. That is, the expression is evaluated as if it were written:

```
(A=B).OR.(C=D)
```

In this expression:

```
.NOT.A<B.AND.A+4/B
```

First A<B is evaluated, then .NOT. A<B. Then 4/B is evaluated and this added to A. Then the first expression is ANDed with the second. That is, the expression is evaluated as if it were written:

```
(.NOT.(A<B)).AND.(A+(4/B))
```

4.4.3 True and false. For expression evaluation, the value of true is 1, false, 0. (Note that the IF pseudo-op interprets any non-zero value as true.)

4.4.4 Logical operators. .RES. unconditionally resets any overflow error in an operand expression. The shift operators shift their first argument right or left by the number of bit positions given in the second argument. Zeros shift into vacated bit positions. The negative (two's complement) of an expression may be formed by preceding it with a minus sign. The one's complement of an expression may be formed by preceding it with the .NOT. operator.

4.4.5 $. The symbol $ represents the value of the program counter of the current instruction. In relative addressing, the program counter must be subtracted from the label if a branch is to be made to the label address.

Example:

```
JR LOOP-$
```

jumps relative to label LOOP.

For a JR on <condition> or a DJNZ the assembler issues an out of range (R) error if and only if the operand expresssion evaluates to >127 or <-128. This introduces the anomaly that JR LOOP is legal as far as the assembler is concerned if the address of LOOP is <128. That is, if the address of LOOP is at, say, 0, and at address 1000H is the instruction JR LOOP-$, even though the jump exceeds 127 bytes, the assembler will not catch the error. But, on execution, the program will not make the jump to LOOP.

4.4.6 Memory addresses. Enclosing an expression completely in parentheses indicates a memory address. In instructions such as LD A,(nn), where nn is a literal address, an expression consisting of symbols and operators may be used as the literal address within the parentheses.

4.5 Op codes

That part of the source instruction that specifies the operation to be performed on the operands is called the op code. There are 74 op codes, 25 operand key words and 643 legitimate combinations of op codes and operands in the Z80 instruction set. The full set of these op codes is summarized in the Z80 CPU Technical Manual and fully described in the Z80 Assembly Language Programming Manual, referred to earlier. Both are published by Zilog Publications, Zilog, Inc., Cupertino, California. (See Section 7 for a summary of the op codes.)

4.6 Pseudo ops

Pseudo-ops do not generate machine instructions; instead, they direct the assembler to do something. EXASM recognizes several pseudo-ops which appear in the op code field of a source statement. Labels for these source lines are optional for all pseudo-ops except two (EQU and DEFL). Pseudo-ops do not necessarily generate object code, but can reserve bytes or can cause certain values to be loaded into certain bytes. However, pseudo-ops always cause some action in the assembler. The assembler recognizes these pseudo-ops:

4.6.1 Data Generation

DEFB/DEFM/DB - define the contents of a byte or bytes located at the current program counter address. DB, DEFB and DEFM are synonymous. Here is the format:

```
<label> DEFB   n[,n,n...]
```

where n is an eight bit value that may be an expression, or a string. DEFB will not generate more than 255 bytes of data. If the value of the expression is greater than eight bits, that is, >255 or <-128, a warning is flagged.

Example:

```
        DEFB    3               ;generates byte of 3
CR      EQU     0DH             ;defines CR
        DEFB    CR              ;generates a byte of CR
                                ;(defined as 0DH)
```

Multiple operands may be used, separated by delimiters.

Example:

```
        DEFB    'HI',233/2+4,'I''M HAL',168
```

Note the use of two apostrophes in "I'M." Two contiguous apostrophes embedded in a string expression generate the ASCII code for one apostrophe. This convention is used because a single apostrophe is construed by the assembler to be a string delimiter.

Example:

```
        DEFB    1,,,2,,3,,4,,,5
```

is the same as:

```
        DEFB    1,2,3,4,5
```

which shows that multiple delimiters are accepted, although one has the same effect.

```
HIMSG:  DEFM 'HI!'              ;picks up message HI!
                                ;and stores it
```

As previously explained, if you want to put quotes into the message, use the apostrophe key twice.

Example:

```
QUOTE:  DEFM 'HE SAID ''HI!'''
```

This produces: HE SAID 'HI!'

There is a simple short-hand way of defining multiple blocks of bytes using DB instructions. These are called ("reps," for repetitions of code) and are used within angle brackets. Suppose you wished to define these bytes:

```
1,1,1,5,5,5,5,5,5,10,10,10,10
```

This is three 1's, six 5's and four 10's. You could do it this way:

```
        DEFB    <3,1,>,<6,5,>,<4,10,>
```

Note that the trailing comma must be present within each pair of brackets. These may be nested up to five deep. A general form for this multiple usage is:

```
<label> DB   <n,W,[<o,X,>,][<p,Y,[<q,Z,>,]>,]...>
```

where n, o, p and q are number of iterations and W, X, Y and Z are numeric literals (any one of which could also be an alpha-numeric literal if enclosed in apostrophes, as 'W' or 'X'). Notice that each iteration has a trailing comma which must be present. A few assembled source statements (including three that cause errors) show this use:

```
                                EXIDY Z80 ASSEMBLER V x.x PAGE   1
 ADDR   OBJECT      ST #

'0000   00000000    0007          DEFB      <5,0,>
        00
'000E   00484901    0008          DEFM      <4,0,'HI',<2,1,>,>
        01004849
        01010048
        49010100
        48490101
'0054   46524F47    0011          DB        <3,'FROGS',<2,'TOADS',>,'CICADA',
        53544F41
        4453544F
        41445343
        49434144
        4146524F
        4753544F
        41445354
        4F414453
        43494341
        44414652
        4F475354
        4F414453
        544F4144
        53434943
        414441
'011A                 0018          DEFB      <257,0,>
****  ERROR CODE = H, REP ERR  ****************************************
'011A                 0019          DB        <130,0,1>
****  ERROR CODE = G, UNBALANCED REP ("<",">") ************************
'011A                 0020          DEFM      <1,<1,<1,<1,<1,<1,0,>,>,>,>,>,>
****  ERROR CODE = H, REP ERR  ****************************************
```

The first statement, DEFB, produces in the object code five bytes of 0: 00 00 00 00 00.

The second statement, DEFM, produces four times the following: one byte of zero followed by two bytes of "H" and "I" followed each time by two bytes of 1, that is:

```
0,H,I,1,1,0,H,I,1,1,0,H,I,1,1,0,H,I,1,1
```

The third statement, DB, produces three times the word "FROGS," followed each time by two "TOADS" and one "CICADA," thus:

```
FROGSTOADSTOADSCICADA
FROGSTOADSTOADSCICADA
FROGSTOADSTOADSCICADA
```

The first error is caused by trying to generate more than 255 bytes of code. The second by leaving off the trailing comma. (If the comma were added, however, an H error would be caused by again trying to generate more than 255 bytes of code.) The last error is caused by nesting too deep.

Only the first four bytes of the object code are shown in the assembly listing when you use the G option with DB/DEFB/DEFM.

DEFW - defines the contents of a two-byte word. The least significant byte of the value nn is loaded at the program counter address. The most significant byte is loaded at program counter plus one. These two bytes together comprise what is termed a "word," having this format:

```
<label> DEFW   <expr>
```

where <expr> is a sixteen bit value or label.

```
PBFR    DEFW   BFR          ;The least significant
                            ;byte of the value of
                            ;BFR is loaded into the
                            ;byte pointed to by
                            ;PBFR and the most
                            ;significant byte is
                            ;loaded into PBFR plus
                            ;one.
```

DEFW supports multiple operands separated by single commas.

Example:

```
            DEFW    1,SYM,XSYM,27+3/455,SYM-12,'HI'
```

DEFS - defines a space of RAM without initalizing it with values. This pseudo-op reserves <expr> bytes of memory starting at the current program counter value. Here is the format:

```
<label> DEFS   <expr>
```

where <expr> is a sixteen bit value or absolute expression. A label used in the operand field of a DEFS statement must be defined before the DEFS statement appears.

```
            BFR     DEFS   200      ;reserve 200 bytes of
                                     ;storage.
```

The DEFS nn statement is the same as an ORG $+nn statement, where $ is the value of the program counter. That is, these two statements produce the same amount of space:

```
DEFS 100
```

and

```
ORG  $+100
```

4.6.2 Source control

IF - defines conditional assembly. If the expression nn is true (non-zero), the IF pseudo-op is ignored. If the expression is false (zero) the assembly of subsequent statements up to the matching ENDIF statement is disabled as if it were not in the source module. The IF pseudo-op cannot be nested. Here is the format:

```
<label> IF   nn
```

where nn is a sixteen bit value.

ENDIF - signals the end of a conditional assembly and reenables assembly of subsequent statements. Here is the format:

```
ENDIF
```

Example:

```
NOASM   EQU     0
        IF      NOASM
        DEFM    'HI THERE'
        ENDIF
```

As long as NOASM has value 0 (false), nothing from the IF statement to the ENDIF statement assembles. That is, in this case, the DEFM statement is not assembled. If NOASM has a value other than 0 (that is, it is true), then assembly does not skip to the ENDIF statement, and statements after the IF statement are assembled. So if the 0 in the EQU statement is changed to a 1, then the DEFM statement is assembled.

INCLUDE - allows source statements from another input file to be included within the body of the given program. If the INCLUDE file cannot be properly opened, then assembly aborts. The source module to be included must not end with an END pseudo-op (because this would terminate assembly). The INCLUDE pseudo-op may not be nested.

```
INCLUDE <filename[.<type>]>
```

where <filename> may be up to eight characters and <type> may be up to three letters. If not specified, <type> defaults to .ASM.

4.6.3 Object control

PSECT - defines a program section as absolute or relocatable. If used, this pseudo-op should appear before any source lines can be assembled into object code and should appear only once in any source module. If not included in a source module, the module is assumed relocatable. It has the following format:

```
<label> PSECT  <opr>
```

where <opr> is either ABS (for an absolute module) or REL (for a relocatable module).

ORG - sets the program counter to the value specified. When used in an absolute module before any source code is assembled into an object code, ORG determines the starting address for the program. In a relocatable program, ORG provides an offset to the base address given when loaded. There may be more than one ORG pseudo-op in a source module. If a source module does not contain ORG pseudo-ops, the program counter is set to zero at the beginning of the assembly. It has the following format:

```
<label> ORG    <expr>
```

where <expr> is a sixteen bit value or expression which is Pass 1 defined.

```
<label> ORG    200H              ;this    sets    program
                                 ;counter to 200H
```

END - defines the last line of the program or module in the following format:

```
<label> END
```

NAME - defines the name of the program (source and object). The name is placed in the heading of the assembly listing and in the first record of the object module. If a NAME pseudo-op does not appear in the module it defaults to six blanks. As with all symbols, NAME may be one to six characters in length. Here is the format:

```
<label> NAME  <string>
```

Here, up to six characters can define the name of the program. If longer than six characters, then it is truncated to the first six characters.

```
NAME    MYPROG                   ;the title MYPROG is
                                 ;now placed in the
                                 ;assembly listing and
                                 ;in the first record of
                                 ;the object module.
```

4.6.4 Listing control

Listing control (assembler directives) are pseudo-ops modifying the assembly listing format. They are not printed with the assembly listing, but are assigned statement numbers. The following assembler directives modify the assembly listing format:

EJECT - causes a printer to eject a page of a listing.

TITLE - causes a printer to eject a page and prints a heading. It has the following format:

```
TITLE   s
```

where s is a string of ASCII characters whose length may not exceed the default line length minus 53. (That is, a standard 132 character print line allows up to 79 characters in the title.) Anything beyond that length causes this warning message:

```
** WARNING CODE = H, TITLE TOO LONG  **********
```

The string s need not be enclosed within quotes.

PAGE - causes the next page number in the heading to be set to the value specified. It has the following format:

```
PAGE   x
```

where x is a value of up to four decimal digits.

LIST - causes an assembly listing to begin.

NLIST - causes an assembly listing to stop until the next LIST directive is found (if any).

LIST and NLIST allow optional activation and deactivation. (See Customizing EXASM.)

4.6.4.1 LIST and NLIST with operands. You may use optional character strings as operands with LIST and NLIST. If you do not use options, then NLIST causes the listing to suspend until the next LIST is encountered. If you use the options, then these two function slightly differently. Rather than stopping the listing, NLIST causes option disable. Rather than resuming listing, LIST causes option assertion. These options alter listing. Options may be strung together. These are the options:

G - Don't print text.

W - Don't print warnings.

E - "Ecology" (suppression of form feeds and ejects).

Examples:

```
        LIST GW
        NLIST GW
```

Example:

```
EXAMPL                                  EXIDY Z80 ASSEMBLER V 2.1  PAGE
  ADDR  OBJECT    ST # SOURCE STATEMENT

                  0001          NAME    EXAMPL
                  0002                          ;
                  0003          LIST    G       ;DO NOT LIST GENERATED TEXT
 '0000  49462054  0004          DEFB    'IF THE OBJECT FOR THIS LISTS, TROUBLE'
                  0005                          ;ONLY FIRST FOUR BYTES SHOULD
                  0006                          ;  LIST
                  0007                          ;
                  0008                          ;
                  0009          NLIST   G       ;LIST GENERATED TEXT
 '0025  54484953  0010          DEFB    'THIS SHOULD LIST THE OBJECT ENTIRELY'
        2053484F
        554C4420
        4C495354
        20544845
        204F424A
        45435420
        454E5449
        52454C59
                  0011                          ;ALL TEXT GENERATED SHOULD
                  0012                          ;  LIST IN EXTRA LINES
                  0013                          ;
                  0014          LIST    W       ;DISABLE WARNINGS
                  0015                          ;
 '0049  01        0016          DEFB    101H    ;SHOULD GIVE NO OVERFLOW WARNING
                  0017                          ;
                  0018          NLIST   W       ;ENABLE WARNINGS
                  0019                          ;
 '004A  01        0020          DEFB    101H    ;SHOULD GIVE OVERFLOW WARNING
** WARNING CODE = V, OVERFLOW    ******************************
                  0021                          ;
                  0022          END


ERRORS=0000

WARNINGS=0001
```

Here, the statement LIST G causes assertion of the option. The option in this case is to <u>not</u> list generated text, so LIST G cuts off text after the fourth byte, as you can see by the object code associated with statement 0004. With the statement NLIST G, the option is suppressed, and text is generated as usual, as seen in the object code for statement 0010.

4.6.5 Symbol control

EQU - assigns a value to a label. The label cannot be defined by an EQU pseudo-op or by appearing in the label field of another source statement in the source module. If a global symbol is defined by an EQU (as seen below), then the value of the global symbol is relocated when linked even though it appears as a constant in the EQU. Here is the format:

```
<label> EQU     <expr>
```

where <expr> is the value.

Example:

```
CONST   EQU     7               ;The value of CONST is 7
```

Labels used in the operand field of an EQU statement must be defined in previous source code. Thus, the following three statements would not be permitted:

```
A       EQU     B
B       EQU     C
C       EQU     0FFFH
```

These, however, are valid:

```
C       EQU     0FFFH
B       EQU     C
A       EQU     B
```

DEFL - defines a label. It sets the value of a label to <expr> and may be used repeatedly for the same label within a module. DEFL is similar in function to EQU but can be multiply used for a particular label. Here is the format:

```
<label> DEFL    <expr>
```

where <expr> is a sixteen bit value or expression.

Example:

```
CURNBR  DEFL    0               ;the value of current #
                                ;is zero for this part
                                ;of the assembly

CURNBR  DEFL    1               ;the value is now one
                                ;in this part of the
                                ;assembly
```

4.6.6 Linking Control

The following pseudo-ops are used to declare a symbol's scope as global and identify the symbol as internal or external. The GLOBAL pseudo-op is the historical ancestor of the other INT/EXT-type pseudo-ops.

The INT/EXT method of symbol reference gives the advantage of error checking for external labels which are accidentally locally defined. If also checks that internal names are spelled correctly.

If a symbol is referenced in a module and is not defined in that module, it must be an external symbol that can be found in a global statement in another module. Conversely, if the global symbol is defined in the module, then it is an internal symbol. Here is the format:

```
<label> GLOBAL  <symbol>
```

Example:

```
        GLOBAL   XSYM          ;This declares XSYM
                               ;global
```

In other assemblers, the GLOBAL pseudo-op is the only pseudo-op used to specify both internal and external global symbols. It may also be so used in EXASM. Whereas elsewhere no differentiation can be made between global externals and internals, here the INT/EXT pseudo-ops may be used in place of GLOBAL.

These three may be used in place of the GLOBAL pseudo-op to specify an external global symbol:

EXTERNAL, EXTERN, EXT

These four may be used in place of the GLOBAL pseudo-op to specify an internal global symbol:

INTERNAL, INTERN, INT, PUBLIC

You may use any of the forms interchangeably.

The advantage of using these ops is that error checking is performed. Here are examples, together with their associated error messages:

```
          EXT     XSYM
*****  ERROR CODE = J, EXT LOCALLY DEFINED  ********************

          INT     ISYM
*****  ERROR CODE = K, INT NOT DEFINED  ************************
```

5 LISTING

5.1 Format

Print file headings look like this:

```
<name>    <title>      EXIDY Z80 ASSEMBLER version x.x  PAGE   n
  ADDR  OBJECT    ST # SOURCE STATEMENT
```

1. The first six characters are the name. They come from the NAME statement.

2. Three blanks follow.

3. Then comes the TITLE (which must conform to TITLE length limitations, described in 4.6.4).

4. Then follows the assembler message.

5. The last item on the first line is the page number. This number is the current page count, unless changed by the PAGE pseudo-op (4.6.4).

6. On the next line are the titles for address, object code, statement number, and source statement.

An apostrophe to the left of an address means that address is relocatable. An apostrophe after the object code means it will be relocated as needed by EXLINK. A trailing asterisk after the object code signifies an external global reference.

See section 5.3 for example listing.

5.2 Error Messages

When an error occurs during assembly, it either causes an abort error condition or generates an error message in the listing. All error messages are designated by a single alpha character. Assembler errors are one of the following types:

5.2.1 ABORT. An error stopping the assembly of a program or module. There are three abort errors. When either occurs, control returns to CP/M with one of these messages output to the console:

```
****ABORT ERROR = Z, OPR REQUESTED ABORT **********************
```

This occurs when the operator presses CONTROL C during assembly.

```
****ABORT ERROR = F, SYMBOL TABLE FULL ************************
```

The symbol table is full, indicating more symbols have been defined than the symbol table can accommodate.

```
****ABORT ERROR = Y, SRC/PRN/OBJ FILES SAME ********************
```

The command specified the same name for two or more files. For example, this command would cause the error:

```
A>EXASM A.ASM,A.ASM
```

5.2.2 MESSAGE

An error or warning that does not stop the assembly of a program or module produces a message that prints in the listing (print file) inserted immediately following the incorrect statement. A single letter abbreviation represents one of these messages. These messages appear on the console together with the statement that caused the problem, as:

```
'0067     0048        LC     (HL),A
*****  ERROR CODE = O, OPCODE  ********************************
```

and

```
'006A     1140F4      LD     DE,0F440H; STARTING ADDRESS TO OUTP
**  WARNING CODE = T, TRUNCATED LINE  ************************
```

They also appear at the appropriate place in the listing.

5.2.2.1 ERROR MESSAGES

A - UNBALANCED PARENS. The number of left parentheses must equal the number of right parentheses.

B - INVALID OPERATOR. An operator not allowed by the assembler exists in an expression. This usually refers to a trailing operator.

C - EXPR TOO COMPLICATED. The expression is too complicated for the assembler to evaluate.

D - INVALID DIGIT. An operand in the source statement is a number with an unallowable digit or character.

E - INVALID EXTERNAL. An external symbol is used in an expression with operators, as the operand of an EQU or DEFL pseudo-op or as the operand requiring an eight bit value.

G - UNBALANCED REP ("<",">"). Repetition symbols (left and right angle brackets) not balanced.

H - REP ERR. More than 256 bytes of code generated, or repetitions nested too deep.

I - INVALID OPERAND. An invalid operand or combination of operands exists for this op code.

J - EXT LOCALLY DEFINED. An external global symbol (that is, one named with EXTERNAL, EXTERN or EXT) is given a definition within its module.

K - INT NOT DEFINED. An internal global symbol (one named with INTERNAL, INTERN, INT or PUBLIC) is not defined within its module.

L - LABEL. An invalid character exists in a label or symbol. This error can also occur for expressions when the assembler scans for a symbol.

M - MULTIPLE DEF. A symbol was defined in the label field of the source program more than once.

N - LABEL REQUIRED. An EQU or DEFL pseudo-op is used without a label in the statement.

O - OPCODE. An invalid op code exists in the op code field of the source statement.

P - MULTIPLE PSECTS. The PSECT pseudo-op exists more than once in the same program. More than one PSECT pseudo-op is not allowed in the same program or module. A module must be either relocatable or absolute, never both.

Q - BAD QUOTE. A string expression has unbalanced quotes.

R - OUT OF RANGE. An operand exists out of the range allowed for the given op code. This often occurs for a JR or DJNZ op code when the operand is too large, that is, the target is too far from the JR or DJNZ instruction (>127 or <-128). It also occurs when "-$" is omitted from the operand label.

S - EXPR SYNTAX. An error in an expression exists. This error usually refers to unbalanced parentheses or extra characters in the expression.

U - UNDEF SYMBOL. A symbol used in an operand expression is not defined in the program or module. This occurs when a symbol is defined by and EQU or DEFL in terms of a local symbol that has not appeared in the source module, or when the undefined symbol is referenced as an instruction operand.

X - PARENS TOO DEEP. Parentheses may be nested no more than fifteen deep, although error code C may come up in the ten to fifteen range (depending on how complicated the parenthesized expressions are).

5.2.2.2 WARNING MESSAGES

H - TITLE TOO LONG. The TITLE pseudo-op supports a title no longer than the line length minus 53. Thus, with the default line length of 132, the title could not exceed 79 characters.

T - TRUNCATED LINE. The input statement exceeds the maximum. When the input statement exceeds the maximum the statement is truncated at the maximum permissible character and the rest ignored. Maximum source line length is a function of print line length as specified in location 106H (line length minus 24; see 6.1.4).

V - OVERFLOW. There are two sources for this warning--expression evaluation and the DEFB/DEFM/DB pseudo-ops. An expression, when evaluated, caused an overflow error in the Z80 CPU (that is, the value exceeded a sixteen-bit field). This can occur for any expression involving arithmetic operators. This can be reset with the .RES. operation. The DEFB/DEFM/DB pseudo-ops generate an overflow warning if an operand expression has a value exceeding an eight-bit field size (>255 or <-127).

5.3 Example listing

In this example EXASM is customized for an eighty-column printer. (How to do it is explained in section 6.) We have written this program:

```
        TITLE   *** MULTIPLY.ASM ***
        NAME    PROG
;
;
;
;
;   TWO-BYTE FULL PRECISION MULTIPLY
;
;
;
        GLOBAL  MULT
;
;
;  UPON ENTRY:
;       H-L CONTAINS 2 BYTE BINARY MULTIPLICAND
;       D-E CONTAINS 2 BYTE BINARY MULTIPLIER
;
;  UPON EXIT:
;       H-L CONTAINS HI ORDER 2 BYTES OF 4 BYTE PRODUCT
;       D-E CONTAINS LO ORDER 2 BYTES OF 4 BYTE PRODUCT
;
;       ALL OTHER REGISTERS PRESERVED EXCEPT AF
;
```

```
MULT:
        PUSH    IX              ;PRESERVE IX
        PUSH    BC              ;ALSO BC
        PUSH    HL              ;SAVE MULTIPLICAND
        LD      IX,PROD         ;INDEX REGISTER POINTS
                                ;  TO PROD
        LD      HL,0            ;ZERO H-L
        LD      (PROD),HL       ;INITIALIZE PRODUCT
                                ;  AREA TO 0
        LD      (PROD+2),HL     ;INITIALIZE LO ORDER
                                ;  TOO
        POP     HL              ;RESTORE MULTIPLICAND
        LD      B,16            ;SHIFT OUT 16 TIMES
SHIFT:
        XOR     A               ;CLEAR CARRY
        RR      D               ;ROTATE RIGHT THRU
                                ;  CARRY
        RR      E               ;THRU LO ORDER BYTE
                                ;  TOO
        CALL    ADDHL           ;IF CARRY, ADD
                                ;  MULTIPLICAND
        DJNZ    SHIFT-$         ;IF MORE BITS TO
                                ;  SHIFT, ITERATE
;
        LD      H,(IX+0)        ;ELSE, PUT HI ORDER
                                ;  IN H
        LD      L,(IX+1)        ;NEXT HIGHEST ORDER
                                ;  IN L
        LD      D,(IX+2)        ;AND PUT LO ORDER IN D
        LD      E,(IX+3)        ;LOWEST IN E
        POP     BC              ;RESTORE STACK
        POP     IX              ;
        RET                     ;RETURN TO CALLING
                                ;  PROGRAM
;
;
;  ADD CONTENTS OF H-L TO HI ORDER OF 4 BYTE PRODUCT
;  AREA.  THEN SHIFT PARTIAL PRODUCT RIGHT.
;
;
;
ADDHL:
        PUSH    DE              ;PRESERVE D-E
        PUSH    HL              ;AND H-L
        JP      NC,OVADD        ;JUST SHIFT IF NO
                                ;  CARRY OUT.
        LD      D,(IX+0)        ;GET HI ORDER PRODUCT
                                ;  IN D
        LD      E,(IX+1)        ;GET 2ND HIGHEST IN E
        ADD     HL,DE           ;ADD IN H-L TO HI
                                ;  ORDER PROD
        LD      (IX+0),H        ;PUT SUM BACK
        LD      (IX+1),L        ;  IN HIGH ORDER OF
                                ;  PRODUCT
```

```
OVADD:
        LD      HL,PROD         ;POINT H-L TO PRODUCT
        RR      (HL)            ;ROTATE RIGHT 1ST BYTE
        INC     HL              ;POINT TO 2ND BYTE OF
                                ;  PROD
        RR      (HL)            ;ROTATE 2ND BYTE THRU
                                ;  CARRY
        INC     HL              ;POINT TO 3RD BYTE OF
                                ;  PROD
        RR      (HL)            ;ROTATE 3RD BYTE THRU
                                ;  CARRY
        INC     HL              ;POINT TO 4TH BYTE OF
                                ;  PROD
        RR      (HL)            ;ROTATE 4TH BYTE
        POP     HL              ;RESTORE H-L
        POP     DE              ;RESTORE D-E
        RET                     ;RETURN
;
PROD    DEFS    4
        DEFB    0
```

We wish to assemble MULTIPLY.ASM, put the object file on disk, print the print file on our printer, list the cross-references, and not waste paper.

```
A>EXASM MULTIPLY/LEC
```

After assembly, this object file is produced:

```
$NAME  050111
$MULT  02000012
:20000000DDE5C5E5DD21500021000022500022520OE10610AFCB1ACB1BCD2E
0010F6DD666A
:2000200000DD6E01DD5602DD5E03C1DDE1C9D5E5D24000DD5600DD5E0119DD
7400DD7501C1
:10004000215000CB1E23CB1E23CB1ECB1EE1D1C9DA
:0100540000AB
$0C0000040006000C000F001A0031004143
:00000001FF
```

The listing produced by the assembly is:

```
PROG        *** MULTIPLY.ASM ***              EXIDY Z80 ASSEMBLER V 2.1  PAGE     1
  ADDR  OBJECT    ST # SOURCE STATEMENT

                  0002          NAME    PROG
                  0003 ;
                  0004 ;
                  0005 ;
                  0006 ;
                  0007 ;    TWO-BYTE FULL PRECISION MULTIPLY
                  0008 ;
                  0009 ;
                  0010 ;
                  0011          GLOBAL  MULT
                  0012 ;
                  0013 ;
                  0014 ;  UPON ENTRY:
                  0015 ;        H-L CONTAINS 2 BYTE BINARY MULTIPLICAND
                  0016 ;        D-E CONTAINS 2 BYTE BINARY MULTIPLIER
                  0017 ;
                  0018 ;  UPON EXIT:
                  0019 ;        H-L CONTAINS HI ORDER 2 BYTES OF 4 BYTE PRODUCT
                  0020 ;        D-E CONTAINS LO ORDER 2 BYTES OF 4 BYTE PRODUCT
                  0021 ;
                  0022 ;        ALL OTHER REGISTERS PRESERVED EXCEPT AF
                  0023 ;
'>0000            0024 MULT:
'0000  DDE5       0025          PUSH    IX                ;PRESERVE IX
'0002  C5         0026          PUSH    BC                ;ALSO BC
'0003  E5         0027          PUSH    HL                ;SAVE MULTIPLICAND
'0004  DD215100'  0028          LD      IX,PROD           ;INDEX REGISTER POINTS
                  0029                                    ;  TO PROD
'0008  210000     0030          LD      HL,0              ;ZERO H-L
'000B  225100'    0031          LD      (PROD),HL         ;INITIALIZE PRODUCT
                  0032                                    ;  AREA TO 0
'000E  225300'    0033          LD      (PROD+2),HL       ;INITIALIZE LO ORDER
                  0034                                    ;  TOO
'0011  E1         0035          POP     HL                ;RESTORE MULTIPLICAND
'0012  0610       0036          LD      B,16              ;SHIFT OUT 16 TIMES
'>0014            0037 SHIFT:
'0014  AF         0038          XOR     A                 ;CLEAR CARRY
'0015  CB1A       0039          RR      D                 ;ROTATE RIGHT THRU
                  0040                                    ;  CARRY
'0017  CB1B       0041          RR      E                 ;THRU LO ORDER BYTE
                  0042                                    ;  TOO
'0019  CD2E00'    0043          CALL    ADDHL             ;IF CARRY, ADD
                  0044                                    ;  MULTIPLICAND
'001C  10F6       0045          DJNZ    SHIFT-$           ;IF MORE BITS TO
                  0046                                    ;  SHIFT, ITERATE
                  0047 ;
'001E  DD6600     0048          LD      H,(IX+0)          ;ELSE, PUT HI ORDER
                  0049                                    ;  IN H
'0021  DD6E01     0050          LD      L,(IX+1)          ;NEXT HIGHEST ORDER
                  0051                                    ;  IN L
'0024  DD5602     0052          LD      D,(IX+2)          ;AND PUT LO ORDER IN D
'0027  DD5E03     0053          LD      E,(IX+3)          ;LOWEST IN E
```

```
PROG        *** MULTIPLY.ASM ***              EXIDY Z80 ASSEMBLER V 2.1  PAGE    2
  ADDR  OBJECT    ST # SOURCE STATEMENT

 '002A  C1        0054          POP     BC              ;RESTORE STACK
 '002B  DDE1      0055          POP     IX              ;
 '002D  C9        0056          RET                     ;RETURN TO CALLING
                  0057                                  ;  PROGRAM
                  0058 ;
                  0059 ;
                  0060 ;  ADD CONTENTS OF H-L TO HI ORDER OF 4 BYTE PRODUCT
                  0061 ;  AREA.  THEN SHIFT PARTIAL PRODUCT RIGHT.
                  0062 ;
                  0063 ;
                  0064 ;
 '>002E           0065 ADDHL:
 '002E  D5        0066          PUSH    DE              ;PRESERVE D-E
 '002F  E5        0067          PUSH    HL              ;AND H-L
 '0030  D24000'   0068          JP      NC,OVADD        ;JUST SHIFT IF NO
                  0069                                  ;  CARRY OUT.
 '0033  DD5600    0070          LD      D,(IX+0)        ;GET HI ORDER PRODUCT
                  0071                                  ;  IN D
 '0036  DD5E01    0072          LD      E,(IX+1)        ;GET 2ND HIGHEST IN E
 '0039  19        0073          ADD     HL,DE           ;ADD IN H-L TO HI
                  0074                                  ;  ORDER PROD
 '003A  DD7400    0075          LD      (IX+0),H        ;PUT SUM BACK
 '003D  DD7501    0076          LD      (IX+1),L        ;  IN HIGH ORDER OF
                  0077                                  ;  PRODUCT
 '>0040           0078 OVADD:
 '0040  215100'   0079          LD      HL,PROD         ;POINT H-L TO PRODUCT
 '0043  CB1E      0080          RR      (HL)            ;ROTATE RIGHT 1ST BYTE
 '0045  23        0081          INC     HL              ;POINT TO 2ND BYTE OF
                  0082                                  ;  PROD
 '0046  CB1E      0083          RR      (HL)            ;ROTATE 2ND BYTE THRU
                  0084                                  ;  CARRY
 '0048  23        0085          INC     HL              ;POINT TO 3RD BYTE OF
                  0086                                  ;  PROD
 '0049  CB1E      0087          RR      (HL)            ;ROTATE 3RD BYTE THRU
                  0088                                  ;  CARRY
 '004B  23        0089          INC     HL              ;POINT TO 4TH BYTE OF
                  0090                                  ;  PROD
 '004C  CB1E      0091          RR      (HL)            ;ROTATE 4TH BYTE
 '004E  E1        0092          POP     HL              ;RESTORE H-L
 '004F  D1        0093          POP     DE              ;RESTORE D-E
 '0050  C9        0094          RET                     ;RETURN
                  0095 ;
 '0051            0096 PROD     DEFS    4
 '0055  00        0097          DEFB    0
SYMBOL    VALUE  TYPE  STMT    STATEMENT REFS
------    -----  ----  ----    --------------

ADDHL     '002E        0065    0043
MULT      '0000   INT  0024    0011
OVADD     '0040        0078    0068
PROD      '0051        0096    0079  0033  0031  0028
SHIFT     '0014        0037    0045
```

```
PROG        *** MULTIPLY.ASM ***              EXIDY Z80 ASSEMBLER V 2.1  PAGE
  ADDR  OBJECT    ST # SOURCE STATEMENT


ERRORS=0000

WARNINGS=0000
```

6 Customizing EXASM

EXASM has certain default values. These are found in locations 103, 104, 105 and 106 (hexadecimal) of the EXASM program. They contain the code for, respectively, default control options, default list options, page length and line length. As supplied to you on disk by Exidy, these locations contain, respectively, the bytes 05H, 08H, 34H and 7B hex. These values correspond to:

list	on	(byte 0)
cross reference	off	(byte 0)
object out	on	(byte 0)
form feed	on	(byte 1)
page length	52	(34H)
line width	123 bytes	(7BH)

If you want your EXASM program to have diferent default values you may customize the program by using the S command (Set) of the DDT program that is supplied on your CP/M system disk. Let's say you wished to change the values in these four locations to these values: 07H, 09H, 37H and 80H. Here's how you do it (your input is underlined):

```
A>DDT EXASM.COM

DDT VERS 1.x
NEXT  PC
3000 0100
-S103

0103 05 07
0104 08 09
0104 34 37
0105 7B 50
0107 31 .
```

Now you have a new, slightly different EXASM program. But it exists only in memory. If you save this version on disk, use the CP/M SAVE command this way:

```
A>SAVE 47 EXASM.COM
```

Why 47? Each 100 bytes represents one page. When DDT signed on, it told you that EXASM exists in memory from address location 0100 to 3000. (DDT deals exclusively in hexadecimal numbers.) 30 hex is 48 decimal. The SAVE command saves from location 0100. Since that first 100 bytes represents one page, we subtract 1 from 48 to get 47. With the same file name, the SAVE command overwrites (and replaces) the old file. If you want two versions of EXASM, one the original and one with your modifications, use a different name in the SAVE command.

```
A>SAVE 47 EXASM1.COM
```

As with other CP/M commands, you can specify the drive as part of the file name, with the default to the currently logged drive.

```
A>SAVE 47 B:EXASM.COM
```

After execution of the previous command, you have a new file on drive B called EXASM.COM that contains your modifications, while the original remains unchanged on drive A.

6.1 Default options.

To see why you might want to change contents of memory locations 103, 104, 105 and 106, let's see what they do. Locations 103 and 104 each consist of a two-digit hexadecimal number (so do 105 and 106, but they're handled differently, as we'll see in a moment). This hexadecimal number may be represented by an eight-digit binary number, each digit of which is called a bit. Each bit may be on (1) or off (0). The number five is represented this way:

bit	7	6	5	4	3	2	1	0
	0	0	0	0	0	1	0	1

The binary number 00000101 (or just 101) is the same as the hexadecimal number 05H. In this example, bits 2 and 0 are turned on and the rest are off. Now, the default control options for EXASM are these:

6.1.1 Default control options (location 103)

Bit 0 = list output
Bit 1 = cross-reference output
Bit 2 = object output

For these first three bits, 1=ON, 0=OFF. The unused bits, bit 3 to bit 7, are always 0. With 05H in location 103, the control options default to list and object output. If you wished the default condition to be no output of print file, you would turn off bit 0 by changing it from 1 to 0. (Here is a possible

reason for doing this. You might have no printer and not wish to fill your disk with print files and thus you don't normally wish listings.) This changes the binary number 00000101 into 00000100, or 05H into 04H. Now, to get a disk file listing (.PRN file), you must use the D option. Without the D option, no print files generate to disk. Turn on any of the bits to change the default to that listed; do so by placing a 1 in the appropriate postion, and placing the equivalent hex number into location 103, as described in the previous section.

Similarly, the default list options are these:

6.1.2 Default list control options (location 104)

Bit 0 = suppress generated text printing
Bit 1 = suppress warning messages
Bit 2 = "ecology option" (compressed listing)
Bit 3 = form feed option

Here is the number 08H in binary representation:

bit	7	6	5	4	3	2	1	0
	0	0	0	0	1	0	0	0

This is the normal default setup for list control options. The normal default situation for list control options, then, is bit 3 only turned on, that is, implementing only the form feed option. If your printer does not have form feed capability, you can turn this bit off.

No generated text refers to DEFM, DEFB, DEFW and DB statements. If you wish text in such statements to truncate after the fourth byte, then turn this bit on. (Change the byte in location 104 from 08H to 09H.) With the bit off (the normal default option), this statement

```
        DEFM    'HI, I''M YOUR FRIEND.'
```

produces, on assembly, the following print file:

```
                                EXIDY Z80 ASSEMBLER V x.x PAGE   1
 ADDR   OBJECT      ST #

'0000   48493C20  0001        DEFM    'HI, I''M YOUR FRIEND.'
        49274D20
        594F5552
        20465249
        454E442E

ERRORS=0000

WARNINGS=0000
```

Notice that the object code contains the complete text.

With the bit turned off, however, on assembly, the following print file is produced:

```
                        EXIDY Z80 ASSEMBLER version 2.1   PAGE   1
 ADDR   OBJECT       ST #

'0000   48493C20  0001          DEFM      'HI, I''M YOUR FRIEND.'

ERRORS=0000

WARNINGS=0000
```

Notice that in the object code text in the listing beyond the fourth byte has been truncated.

Similarly, if you wish to default to "ecology option" (paper saving--suppresses form feeds in text), rather than having to always type in /E as an option, turn on bit 2.

6.1.3 Page length (location 105)

The default page length (lines per page) is 52 (34H). You can change this to any value up to 255 (FF) by the method described earlier.

6.1.4 Line length (location 106)

The current default value for column width is 132 (84H). If you have an 80-column printer, you'll want to change this byte to 50H (or any other value). Note that printer line length minus 24 (decimal) is the maximum source line length.

Z80 MICROPROCESSOR INSTRUCTION SET

Z80–CPU INSTRUCTIONS SORTED BY OP-CODE

OBJ CODE	SOURCE STATEMENT
00	NOP
018405	LD BC NN
02	LD (BC) A
03	INC BC
04	INC B
05	DEC B
0620	LD B N
07	RLCA
08	Ex AF AF
09	ADD HL BC
0A	LD A,(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E20	LD C,N
0F	RRCA
102E	DJNZ DIS
118405	LD DE NN
12	LD (DE) A
13	INC DE
14	INC D
15	DEC D
1620	LD D,N
17	RLA
182E	JR DIS
19	ADD HL,DE
1A	LD A (DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E20	LD E N
1F	RRA
202E	JR NZ DIS
218405	LD HL NN
228405	LD (NN) HL
23	INC HL
24	INC H
25	DEC H
2620	LD H N
27	DAA
282E	JR Z DIS
29	ADD HL HL
2A8405	LD HL (NN)
2B	DEC HL
2C	INC L
2D	DEC L
2E20	LD L N
2F	CPL
302E	JR NC DIS
318405	LD SP,NN
328405	LD (NN) A
33	INC SP
34	INC (HL)
35	DEC (HL)
3620	LD (HL),N
37	SCF
382E	JR C DIS
39	ADD HL SP
3A8405	LD A (NN)
3B	DEC SP
3C	INC A
3D	DEC A
3E20	LD A N
3F	CCF
40	LD B B
41	LD B C
42	LD B D
43	LD B E
44	LD B H NN
45	LD B L
46	LD B (HL)
47	LD B A
48	LD C B
49	LD C C
4A	LD C D
4B	LD C E
4C	LD C H
4D	LD C L
4E	LD C (HL)
4F	LD C A
50	LD D B
51	LD D C
52	LD D D
53	LD D E
54	LD D H
55	LD D L
56	LD D (HL)
57	LD D A
58	LD E,B
59	LD E,C
5A	LD E D
5B	LD E E
5C	LD E H
5D	LD E,L
5E	LD E,(HL)
5F	LD E,A
60	LD H,B
61	LD H C
62	LD H,D
63	LD H,E
64	LD H H
65	LD H L
66	LD H (HL)
67	LD H A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L E
6C	LD L,H
6D	LD L,L
6E	LD L,(HL)
6F	LD L,A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A B
79	LD A C
7A	LD A,D
7B	LD A E
7C	LD A H
7D	LD A L
7E	LD A,(HL)
7F	LD A,A
80	ADD A B
81	ADD A,C
82	ADD A D
83	ADD A E
84	ADD A,H
85	ADD A L
86	ADD A (HL)
87	ADD A A
88	ADC A,B
89	ADC A C
8A	ADC A D
8B	ADC A E
8C	ADC A H
8D	ADC A L
8E	ADC A (HL)
8F	ADC A A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A B
99	SBC A C
9A	SBC A D
9B	SBC A E
9C	SBC A H
9D	SBC A L
9E	SBC A (HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C28405	JP NZ NN
C38405	JP NN
C48405	CALL NZ NN
C5	PUSH BC
C620	ADD A N
C7	RST 0
C8	RET Z
C9	RET
CA8405	JP Z NN
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	HRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB26	SLA (HL)
CB27	SLA A
CB28	SHA B
CB29	SHA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SHA (HL)
CB2F	SHA A
CB38	SHL B
CB39	SHL C
CB3A	SHL D
CB3B	SRL E
CB3C	SHL H
CB3D	SHL L
CB3E	SRL (HL)
CB3F	SRL A
CB40	BIT 0,B
CB41	BIT 0,C
CB42	BIT 0,D
CB43	BIT 0,E
CB44	BIT 0,H
CB45	BIT 0,L
CB46	BIT 0,(HL)
CB47	BIT 0,A
CB48	BIT 1,B
CB49	BIT 1,C
CB4A	BIT 1,D
CB4B	BIT 1,E
CB4C	BIT 1,H
CB4D	BIT 1,L
CB4E	BIT 1,(HL)
CB4F	BIT 1,A
CB50	BIT 2,B
CB51	BIT 2,C
CB52	BIT 2,D
CB53	BIT 2,E
CB54	BIT 2,H
CB55	BIT 2,L
CB56	BIT 2,(HL)
CB57	BIT 2,A
CB58	BIT 3,B
CB59	BIT 3,C
CB5A	BIT 3,D
CB5B	BIT 3,E
CB5C	BIT 3,H
CB5D	BIT 3,L
CB5E	BIT 3,(HL)
CB5F	BIT 3,A
CB60	BIT 4,B
CB61	BIT 4,C
CB62	BIT 4,D
CB63	BIT 4,E
CB64	BIT 4,H
CB65	BIT 4,L
CB66	BIT 4,(HL)
CB67	BIT 4,A
CB68	BIT 5,B
CB69	BIT 5,C
CB6A	BIT 5,D
CB6B	BIT 5,E
CB6C	BIT 5,H
CB6D	BIT 5,L
CB6E	BIT 5,(HL)
CB6F	BIT 5,A
CB70	BIT 6,B
CB71	BIT 6,C
CB72	BIT 6,D
CB73	BIT 6,E
CB74	BIT 6,H
CB75	BIT 6,L
CB76	BIT 6,(HL)
CB77	BIT 6,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
CB7E	BIT 7,(HL)
CB7F	BIT 7,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H
CB85	RES 0,L
CB86	RES 0,(HL)
CB87	RES 0,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB8E	RES 1,(HL)
CB8F	RES 1,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB96	RES 2,(HL)
CB97	RES 2,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CB9E	RES 3,(HL)
CB9F	RES 3,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBA6	RES 4,(HL)
CBA7	RES 4,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBAE	RES 5,(HL)
CBAF	RES 5,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBB6	RES 6,(HL)
CBB7	RES 6,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
CBBE	RES 7,(HL)
CBBF	RES 7,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBC6	SET 0,(HL)
CBC7	SET 0,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBCE	SET 1,(HL)
CBCF	SET 1,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBE6	SET 4,(HL)
CBE7	SET 4,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBEE	SET 5,(HL)
CBEF	SET 5,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBF6	SET 6,(HL)
CBF7	SET 6,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CBFE	SET 7,(HL)
CBFF	SET 7,A
CC8405	CALL Z,NN
CD8405	CALL NN
CE20	ADC A N
CF	RST 8
D0	RET NC
D1	POP DE
D2840[illegible]	JP NC NN
D320	OUT (N),A
D48405	CALL NC NN
D5	PUSH DE
D620	SUB N
D7	RST 10H
D8	RET C
D9	EXX
DA840[illegible]	JP C,NN
DB20	IN A,N
DC840[illegible]	CALL C,NN
DD09	ADD IX,BC
DD19	ADD IX,DE
DD218405	LD IX,NN
DD228405	LD (NN),IX
DD23	INC IX
DD29	ADD IX,IX
DD2A8405	LD IX,(NN)
DD2B	DEC IX
DD3405	INC (IX+d)
DD3505	DEC (IX+d)
DD360520	LD (IX+d),N
DD39	ADD IX,SP
DD4605	LD B,(IX+d)
DD4E05	LD C,(IX+d)
DD5605	LD D,(IX+d)
DD5E05	LD E,(IX+d)
DD6605	LD H,(IX+d)
DD6E05	LD L,(IX+d)
DD7005	LD (IX+d),B
DD7105	LD (IX+d),C
DD7205	LD (IX+d),D
DD7305	LD (IX+d),E
DD7405	LD (IX+d),H
DD7505	LD (IX+d),L
DD7705	LD (IX+d),A
DD7E05	LD A,(IX+d)
DD8605	ADD A,(IX+d)
DD8E05	ADC A,(IX+d)
DD9605	SUB (IX+d)
DD9E05	SBC A,(IX+d)
DDA605	AND (IX+d)
DDAE05	XOR (IX+d)
DDB605	OR (IX+d)
DDBE05	CP (IX+d)
DDE1	POP IX
DDE3	EX (SP),IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP,IX
DDCB0506	RLC (IX+d)
DDCB050E	RRC (IX+d)
DDCB0516	RL (IX+d)
DDCB051E	RR (IX+d)
DDCB0526	SLA (IX+d)
DDCB052E	SRA (IX+d)
DDCB053E	SRL (IX+d)
DDCB0546	BIT 0,(IX+d)
DDCB054E	BIT 1,(IX+d)
DDCB0556	BIT 2,(IX+d)
DDCB055E	BIT 3,(IX+d)
DDCB0566	BIT 4,(IX+d)
DDCB056E	BIT 5,(IX+d)
DDCB0576	BIT 6,(IX+d)
DDCB057E	BIT 7,(IX+d)
DDCB0586	RES 0,(IX+d)
DDCB058E	RES 1,(IX+d)
DDCB0596	RES 2,(IX+d)
DDCB059E	RES 3,(IX+d)
DDCB05A6	RES 4,(IX+d)
DDCB05AE	RES 5,(IX+d)
DDCB05B6	RES 6,(IX+d)
DDCB05BE	RES 7,(IX+d)
DDCB05C6	SET 0,(IX+d)
DDCB05CE	SET 1,(IX+d)
DDCB05D6	SET 2,(IX+d)
DDCB05DE	SET 3,(IX+d)
DDCB05E6	SET 4,(IX+d)
DDCB05EE	SET 5,(IX+d)
DDCB05F6	SET 6,(IX+d)
DDCB05FE	SET 7,(IX+d)
DE20	SBC A N
DF	RST 18H
E0	RET PO
E1	POP HL
E28405	JP PO NN
E3	EX (SP),HL
E48405	CALL PO NN
E5	PUSH HL
E620	AND N
E7	RST 20H
E8	RET PE
E9	JP (HL)
EA8405	JP PE NN
EB	EX DE,HL
EC8405	CALL PE NN
ED40	IN B,(C)
ED41	OUT (C),B
ED42	SBC HL,BC
ED438405	LD (NN),BC
ED44	NEG
ED45	RETN
ED46	IM 0
ED47	LD I,A
ED48	IN C,(C)
ED49	OUT (C),C
ED4A	ADC HL,BC
ED4B8405	LD BC,(NN)
ED4D	RETI
ED50	IN D,(C)
ED51	OUT (C),D
ED52	SBC HL,DE
ED538405	LD (NN),DE
ED56	IM 1
ED57	LD A,I
ED58	IN E,(C)
ED59	OUT (C),E
ED5A	ADC HL,DE
ED5B8405	LD DE,(NN)
ED5E	IM 2
ED60	IN H,(C)
ED61	OUT (C),H
ED62	SBC HL,HL
ED67	RRD
ED68	IN L,(C)
ED69	OUT (C),L
ED6A	ADC HL,HL
ED6F	RLD
ED72	SBC HL,SP
ED738405	LD (NN),SP
ED78	IN A,(C)
ED79	OUT (C),A
ED7A	ADC HL,SP
ED7B8405	LD SP,(NN)
EDA0	LDI
EDA1	CPI
EDA2	INI
EDA3	OUTI
EDA8	LDD
EDA9	CPD
EDAA	IND
EDAB	OUTD
EDB0	LDIR
EDB1	CPIR
EDB2	INIR
EDB3	OTIR
EDB8	LDDR
EDB9	CPDR
EDBA	INDR
EDBB	OTDR
EE20	XOR N
EF	RST 28H
F0	RET P
F1	POP AF
F28405	JP P,NN
F3	DI
F48405	CALL P,NN
F5	PUSH AF
F620	OR N
F7	RST 30H
F8	RET M
F9	LD SP,HL
FA8405	JP M,NN
FB	EI
FC8405	CALL M,NN
FD09	ADD IY,BC
FD19	ADD IY,DE
FD218405	LD IY,NN
FD228405	LD (NN),IY
FD23	INC IY
FD29	ADD IY,IY
FD2A8405	LD IY,(NN)
FD2B	DEC IY
FD3405	INC (IY+d)
FD3505	DEC (IY+d)
FD360520	LD (IY+d),N
FD39	ADD IY,SP
FD4605	LD B,(IY+d)
FD4E05	LD C,(IY+d)
FD5605	LD D,(IY+d)
FD5E05	LD E,(IY+d)
FD6605	LD H,(IY+d)
FD6E05	LD L,(IY+d)
FD7005	LD (IY+d),B
FD7105	LD (IY+d),C
FD7205	LD (IY+d),D
FD7305	LD (IY+d),E
FD7405	LD (IY+d),H
FD7505	LD (IY+d),L
FD7705	LD (IY+d),A
FD7E05	LD A,(IY+d)
FD8605	ADD A,(IY+d)
FD8E05	ADC A,(IY+d)
FD9605	SUB (IY+d)
FD9E05	SBC A,(IY+d)
FDA605	AND (IY+d)
FDAE05	XOR (IY+d)
FDB605	OR (IY+d)
FDBE05	CP (IY+d)
FDE1	POP IY
FDE3	EX (SP),IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP,IY
FDCB0506	RLC (IY+d)
FDCB050E	RRC (IY+d)
FDCB0516	RL (IY+d)
FDCB051E	RR (IY+d)
FDCB0526	SLA (IY+d)
FDCB052E	SRA (IY+d)
FDCB053E	SRL (IY+d)
FDCB0546	BIT 0,(IY+d)
FDCB054E	BIT 1,(IY+d)
FDCB0556	BIT 2,(IY+d)
FDCB055E	BIT 3,(IY+d)
FDCB0566	BIT 4,(IY+d)
FDCB056E	BIT 5,(IY+d)
FDCB0576	BIT 6,(IY+d)
FDCB057E	BIT 7,(IY+d)
FDCB0586	RES 0,(IY+d)
FDCB058E	RES 1,(IY+d)
FDCB0596	RES 2,(IY+d)
FDCB059E	RES 3,(IY+d)
FDCB05A6	RES 4,(IY+d)
FDCB05AE	RES 5,(IY+d)
FDCB05B6	RES 6,(IY+d)
FDCB05BE	RES 7,(IY+d)
FDCB05C6	SET 0,(IY+d)
FDCB05CE	SET 1,(IY+d)
FDCB05D6	SET 2,(IY+d)
FDCB05DE	SET 3,(IY+d)
FDCB05E6	SET 4,(IY+d)
FDCB05EE	SET 5,(IY+d)
FDCB05F6	SET 6,(IY+d)
FDCB05FE	SET 7,(IY+d)
FE20	CP N
FF	RST 38H

Z80–CPU INSTRUCTIONS SORTED BY MNEMONIC

OBJ CODE	SOURCE STATEMENT
8E	ADC A (HL)
DD8E05	ADC A (IX+d)
FD8E05	ADC A (IY+d)
8F	ADC A A
88	ADC A B
89	ADC A C
8A	ADC A D
8B	ADC A E
8C	ADC A H
8D	ADC A L
CE20	ADC A N
ED4A	ADC HL BC
ED5A	ADC HL DE
ED6A	ADC HL HL
ED7A	ADC HL SP
86	ADD A (HL)
DD8605	ADD A (IX+d)
FD8605	ADD A (IY+d)
87	ADD A A
80	ADD A B
81	ADD A C
82	ADD A D
83	ADD A E
84	ADD A H
85	ADD A L
C620	ADD A N
09	ADD HL BC
19	ADD HL DE
29	ADD HL HL
39	ADD HL SP
DD09	ADD IX BC
DD19	ADD IX DE
DD29	ADD IX IX
DD39	ADD IX SP
FD09	ADD IY BC
FD19	ADD IY DE
FD29	ADD IY IY
FD39	ADD IY SP
A6	AND (HL)
DDA605	AND (IX+d)
FDA605	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E620	AND N
CB46	BIT 0 (HL)
DDCB0546	BIT 0 (IX+d)
FDCB0546	BIT 0 (IY+d)
CB47	BIT 0 A
CB40	BIT 0 B
CB41	BIT 0 C
CB42	BIT 0 D
CB43	BIT 0 E
CB44	BIT 0 H
CB45	BIT 0 L
CB4E	BIT 1 (HL)
DDCB054E	BIT 1 (IX+d)
FDCB054E	BIT 1 (IY+d)
CB4F	BIT 1 A
8C48	BIT 1 B
CB49	BIT 1 C
CB4A	BIT 1 D
CB4B	BIT 1 E
CB4C	BIT 1 H
CB4D	BIT 1 L
CB56	BIT 2 (HL)
DDCB0556	BIT 2 (IX+d)
FDCB0556	BIT 2 (IY+d)
CB57	BIT 2 A
CB50	BIT 2 B
CB51	BIT 2 C
CB52	BIT 2 D
CB53	BIT 2 E
CB54	BIT 2 H
CB55	BIT 2 L
CB5E	BIT 3 (HL)
DDCB055E	BIT 3 (IX+d)
FDCB055E	BIT 3 (IY+d)
CB5F	BIT 3 A
CB58	BIT 3 B
CB59	BIT 3 C
CB5A	BIT 3 D
CB5B	BIT 3 E
CB5C	BIT 3 H
CB5D	BIT 3 L
CB66	BIT 4 (HL)
DDCB0566	BIT 4 (IX+d)
FDCB0566	BIT 4 (IY+d)
CB67	BIT 4 A
CB60	BIT 4 B
CB61	BIT 4 C
CB62	BIT 4 D
CB63	BIT 4 E
CB64	BIT 4 H
CB65	BIT 4 L
CB6E	BIT 5 (HL)
DDCB056E	BIT 5 (IX+d)
FDCB056E	BIT 5 (IY+d)
CB6F	BIT 5 A
CB68	BIT 5 B
CB69	BIT 5 C
CB6A	BIT 5 D
CB6B	BIT 5 E
CB6C	BIT 5 H
CB6D	BIT 5 L
CB76	BIT 6 (HL)
DDCB0576	BIT 6 (IX+d)
FDCB0576	BIT 6 (IY+d)
CB77	BIT 6 A
CB70	BIT 6 B
CB71	BIT 6 C
CB72	BIT 6 D
CB73	BIT 6 E
CB74	BIT 6 H
CB75	BIT 6 L
CB7E	BIT 7 (HL)
DDCB057E	BIT 7 (IX+d)
FDCB057E	BIT 7 (IY+d)
CB7F	BIT 7 A
CB78	BIT 7 B
CB79	BIT 7 C
CB7A	BIT 7 D
CB7B	BIT 7 E
CB7C	BIT 7 H
CB7D	BIT 7 L
DC8405	CALL C NN
FC8405	CALL M NN
D48405	CALL NC NN
CD8405	CALL NN
C48405	CALL NZ NN
F48405	CALL P NN
EC8405	CALL PE NN
E48405	CALL PO NN
CC8405	CALL Z NN
3F	CCF
BE	CP (HL)
DDBE05	CP (IX+d)
FDBE05	CP (IY+d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
FE20	CP N
EDA9	CPD
ED89	CPDR
EDA1	CPI
EDB1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD3505	DEC (IX+d)
FD3505	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
102E	DJNZ DIS
FB	EI
E3	EX (SP) HL
DDE3	EX (SP) IX
FDE3	EX (SP) IY
08	EX AF AF'
EB	EX DE HL
D9	EXX
76	HALT
ED46	IM 0
ED56	IM 1
ED5E	IM 2
ED78	IN A,(C)
DB20	IN A,(N)
ED40	IN B (C)
ED48	IN C (C)
ED50	IN D,(C)
ED58	IN E,(C)
ED60	IN H (C)
ED68	IN L (C)
34	INC (HL)
DD3405	INC (IX+d)
FD3405	INC (IY+d)
3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
EDAA	IND
EDBA	INDR
EDA2	INI
EDB2	INIR
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA8405	JP C,NN
FA8405	JP M,NN
D28405	JP NC,NN
C38405	JP NN
C28405	JP NZ,NN
F28405	JP P,NN
EA8405	JP PE,NN
E28405	JP PO,NN
CA8405	JP Z,NN
382E	JR C,DIS
182E	JR DIS
302E	JR NC,DIS
202E	JR NZ,DIS
282E	JR Z,DIS
02	LD (BC),A
12	LD (DE),A
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
3620	LD (HL),N
DD7705	LD (IX+d),A
DD7005	LD (IX+d),B
DD7105	LD (IX+d),C
DD7205	LD (IX+d),D
DD7305	LD (IX+d),E
DD7405	LD (IX+d),H
DD7505	LD (IX+d),L
DD360520	LD (IX+d),N
FD7705	LD (IY+d),A
FD7005	LD (IY+d),B
FD7105	LD (IY+d),C
FD7205	LD (IY+d),D
FD7305	LD (IY+d),E
FD7405	LD (IY+d),H
FD7505	LD (IY+d),L
FD360520	LD (IY+d),N
328405	LD (NN),A
ED438405	LD (NN),BC
ED538405	LD (NN),DE
228405	LD (NN),HL
DD228405	LD (NN),IX
FD228405	LD (NN),IY
ED738405	LD (NN),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E05	LD A,(IX+d)
FD7E05	LD A,(IY+d)
3A8405	LD A,(NN)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED57	LD A,I
7D	LD A,L
3E20	LD A,N
46	LD B,(HL)
DD4605	LD B,(IX+d)
FD4605	LD B,(IY+d)
47	LD B,A
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
0620	LD B,N
ED4B8405	LD BC,(NN)
018405	LD BC,NN
4E	LD C,(HL)
DD4E05	LD C,(IX+d)
FD4E05	LD C,(IY+d)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E20	LD C,N
56	LD D,(HL)
DD5605	LD D,(IX+d)
FD5605	LD D,(IY+d)
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
1620	LD D,N
ED5B8405	LD DE,(NN)
118405	LD DE,NN
5E	LD E,(HL)
DD5E05	LD E,(IX+d)
FD5E05	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E20	LD E,N
66	LD H,(HL)
DD6605	LD H,(IX+d)
FD6605	LD H,(IY+d)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
2620	LD H,N
2A8405	LD HL,(NN)
218405	LD HL,NN
ED47	LD I,A
DD2A8405	LD IX,(NN)
DD218405	LD IX,NN
FD2A8405	LD IY,(NN)
FD218405	LD IY,NN
6E	LD L,(HL)
DD6E05	LD L,(IX+d)
FD6E05	LD L,(IY+d)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2F20	LD L,N
ED7B8405	LD SP,(NN)
F9	LD SP,HL
DDF9	LD SP,IX
FDF9	LD SP,IY
318405	LD SP,NN
EDA8	LDD
EDB8	LDDR
EDA0	LDI
EDB0	LDIR
ED44	NEG
00	NOP
B6	OR (HL)
DDB605	OR (IX+d)
FDB605	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
F620	OR N
EDBB	OTDR
EDB3	OTIR
ED79	OUT (C),A
ED41	OUT (C),B
ED49	OUT (C),C
ED51	OUT (C),D
ED59	OUT (C),E
ED61	OUT (C),H
ED69	OUT (C),L
D320	OUT (N),A
EDAB	OUTD
EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0,(HL)
DDCB0586	RES 0,(IX+d)
FDCB0586	RES 0,(IY+d)
CB87	RES 0,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H
CB85	RES 0,L
CB8E	RES 1,(HL)
DDCB058E	RES 1,(IX+d)
FDCB058E	RES 1,(IY+d)
CB8F	RES 1,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB96	RES 2,(HL)
DDCB0596	RES 2,(IX+d)
FDCB0596	RES 2,(IY+d)
CB97	RES 2,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB9F	RES 3,(HL)
DDCB059F	RES 3,(IX+d)
FDCB059F	RES 3,(IY+d)
CB9F	RES 3,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CBA6	RES 4,(HL)
DDCB05A6	RES 4,(IX+d)
FDCB05A6	RES 4,(IY+d)
CBA7	RES 4,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBAE	RES 5,(HL)
DDCB05AE	RES 5,(IX+d)
FDCB05AE	RES 5,(IY+d)
CBAF	RES 5,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBB6	RES 6,(HL)
DDCB05B6	RES 6,(IX+d)
FDCB05B6	RES 6,(IY+d)
CBB7	RES 6,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBBE	RES 7,(HL)
DDCB05BE	RES 7,(IX+d)
FDCB05BF	RES 7,(IY+d)
CBBF	RES 7,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
C9	RET
D8	RET C
F8	RET M
D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO
C8	RET Z
ED4D	RETI
ED45	RETN
CB16	RL (HL)
DDCB0516	RL (IX+d)
FDCB0516	RL (IY+d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB0506	RLC (IX+d)
FDCB0506	RLC (IY+d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
07	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB051E	RR (IX+d)
FDCB051E	RR (IY+d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA
CB0E	RRC (HL)
DDCB050E	RRC (IX+d)
FDCB050F	RRC (IY+d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
ED67	RRD
C7	RST 0
D7	RST 10H
DF	RST 18H
E7	RST 20H
EF	RST 28H
F7	RST 30H
FF	RST 38H
CF	RST 8
9E	SBC A (HL)
DD9E05	SBC A (IX+d)
FD9E05	SBC A (IY+d)
9F	SBC A A
98	SBC A B
99	SBC A C
9A	SBC A D
9B	SBC A E
9C	SBC A H
9D	SBC A L
DE20	SBC A N
ED42	SBC HL BC
ED52	SBC HL DE
ED62	SBC HL HL
ED72	SBC HL SP
37	SCF
CBC6	SET 0 (HL)
DDCB05C6	SET 0 (IX+d)
FDCB05C6	SET 0 (IY+d)
CBC7	SET 0 A
CBC0	SET 0 B
CBC1	SET 0 C
CBC2	SET 0 D
CBC3	SET 0 E
CBC4	SET 0 H
CBC5	SET 0 L
CBCE	SET 1 (HL)
DDCB05CE	SET 1 (IX+d)
FDCB05CE	SET 1 (IY+d)
CBCF	SET 1 A
CBC8	SET 1 B
CBC9	SET 1 C
CBCA	SET 1 D
CBCB	SET 1 E
CBCC	SET 1 H
CBCD	SET 1 L
CBD6	SET 2 (HL)
DDCB05D6	SET 2 (IX+d)
FDCB05D6	SET 2 (IY+d)
CBD7	SET 2 A
CBD0	SET 2 B
CBD1	SET 2 C
CBD2	SET 2 D
CBD3	SET 2 E
CBD4	SET 2 H
CBD5	SET 2 L
CBD8	SET 3 B
CBDE	SET 3 (HL)
DDCB05DE	SET 3 (IX+d)
FDCB05DE	SET 3 (IY+d)
CBDF	SET 3 A
CBD9	SET 3 C
CBDA	SET 3 D
CBDB	SET 3 E
CBDC	SET 3 H
CBDD	SET 3 L
CBE6	SET 4 (HL)
DDCB05E6	SET 4 (IX+d)
FDCB05E6	SET 4 (IY+d)
CBE7	SET 4 A
CBE0	SET 4 B
CBE1	SET 4 C
CBE2	SET 4 D
CBE3	SET 4 E
CBE4	SET 4 H
CBE5	SET 4 L
CBEE	SET 5 (HL)
DDCB05EE	SET 5 (IX+d)
FDCB05EE	SET 5 (IY+d)
CBEF	SET 5 A
CBE8	SET 5 B
CBE9	SET 5 C
CBEA	SET 5 D
CBEB	SET 5 E
CBEC	SET 5 H
CBED	SET 5 L
CBF6	SET 6 (HL)
DDCB05F6	SET 6 (IX+d)
FDCB05F6	SET 6 (IY+d)
CBF7	SET 6 A
CBF0	SET 6 B
CBF1	SET 6 C
CBF2	SET 6 D
CBF3	SET 6 E
CBF4	SET 6 H
CBF5	SET 6 L
CBFE	SET 7 (HL)
DDCB05FE	SET 7 (IX+d)
FDCB05FE	SET 7 (IY+d)
CBFF	SET 7 A
CBF8	SET 7 B
CBF9	SET 7 C
CBFA	SET 7 D
CBFB	SET 7 E
CBFC	SET 7 H
CBFD	SET 7 L
CB26	SLA (HL)
DDCB0526	SLA (IX+d)
FDCB0526	SLA (IY+d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB052E	SRA (IX+d)
FDCB052E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB053E	SRL (IX+d)
FDCB053E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD9605	SUB (IX+d)
FD9605	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
D620	SUB N
AE	XOR (HL)
DDAE05	XOR (IX+d)
FDAE05	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE20	XOR N

EXLINK is itself relocatable since it finds the user's BDOS and overlays the Command Console Processor (CCP) at the high end of CP/M's memory. That is, while EXLINK is loading, it locates itself in that area in RAM. This feature provides the maximum amount of RAM available for user modules. The diagrams to follow graphically represent these locations. In this respect, EXLINK could be called the Relocating, Relocating Linking Loader!

9 OPERATION

EXLINK is called from CP/M by typing EXLINK on the command line. To use batch mode, type EXLINK followed by a list of file names and a list of up to three options. The two lists are separated by a slash (/). If you specify no file names on the CP/M command line, the program signs on, enters the interactive mode, gives an asterisk (*) as a prompt, and waits for a valid EXLINK command. All lower case input is converted to upper case automatically. All filenames must be alphanumeric.

When EXLINK is called, it immediately fills all the memory into which the user could load his programs, with zeros (from 100 hex to the start of the EXLINK program). This sets all DEFS areas to zero.

9.1 EXLINK interactive mode commands

In these examples, information in square brackets [] is optional user input, and angle brackets <> refer to input as described in the text.

9.1.1 *L [d:]filename[.OBJ] [ZZZZ]

This command finds the .OBJ file with the given file name on the logged in drive (or, if the option d:--for any valid CP/M drive --is specified then on drive d:). The filetype .OBJ may be specified but is the default and only valid filetype. It creates a memory image of the file, and relocates it for the optional starting offset ZZZZ. This is done by one or more of the statements ORG YYYY, or ZZZZ (offset number), as shown here:

```
*L FILE1

*L FILE1 100

*L B:FILE1

*L B:FILE1.OBJ.

*L FILE1.OBJ 200
```

9.1.2 *T

This command prints the current global symbol table.

9.1.3 *E [d:][<filename>][.COM]

This command exits the loader by writing the newly-linked program in memory to a disk COM file (on optional drive d:) starting at memory address 100 hex up to the highest address loaded. The file name is that of the first object module loaded or the user may optionally specify a different name for the COM file to be written by including the <filename> option in the command.

```
*E

*E B:

*E FILE2

*E FILE2.COM

*E B:FILE2.COM
```

After the E command has written a .COM file, the message FILENAME.COM SAVED, RECS WRITTEN=XX" appears. XX is a hexidecimal number referring to how many 128-byte CP/M records comprise the .COM file.

9.1.4 *Q

Both this command and Control C quit EXLINK and return to CP/M without writing a COM file. This is useful to abort an EXLINK operation.

The command line may include up to three options, listed in any order, separated by commas (see 9.2 for details). The options E and T may be entered in interactive mode as commands. Interactive mode is indicated by the EXLINK prompt * when either all the command line has been exhausted successfully (and no E option was found), or a non-fatal error has occurred. Unrecognized options are ignored.

If a list of batch files is given on the command line and the E option is not specified in the options list, EXLINK returns to the interactive mode with the asterisk (*) prompt after all the batch files are loaded. Then, additional OBJ files may be loaded interactively before exiting. This also occurs if any non-fatal errors occur during batch mode operations.

9.2 Batch mode options

The general form of the CP/M command line when using EXLINK in batch mode is:

a>EXLINK [<filename1>] [,filename2>][,<filename3>]...[/options]

<filename1> (the first module's name) is the file name the E command uses when it creates the COM file, provided no other file name is specified either on the CP/M command line (with the E option) or interactively (with the E command). File names on the CP/M command line are delimited by commas. Your command line may contain as many characters as will fit in two lines (up to a total of 128 characters). The OBJ files in the list are ordinarily accessed on the drive currently logged on, unless you specify a drive using the CP/M convention d:<filename> for file <filename>.OBJ on drive d:. The filetype .OBJ may be specified, but is assumed if it is omitted. Only .OBJ files are accepted. The options list follows the file name list and is separated from the list of files by a slash (/). It may include up to three options, listed in any order, separated by commas.

The command line options are:

9.2.1 A=XXXX [SSSS]

XXXX represents the starting offset to be added to the ORG address (if any) of the first OBJ module, and SSSS represents the optional starting address of the global symbol table. As the symbols are added to the table at the given address, the table expands to a lower location or "grows down". As the symbol table expands, then, the RAM available for programs shrinks in size.

The starting address default is 0 if this option is not used. If SSSS is not specified, the symbol table starts at the highest RAM address available just below EXLINK's code area. The symbol table option should not ordinarily be used since the symbol table is automatically positioned in the best possible place for most applications. If the option is used however, care must be taken to prevent the table from "walking" on the EXLINK program, BDOS, page 0 of memory or the users program which is being linked.

9.2.2 E[d:][<filename>][.COM]

This automatically exits EXLINK by writing a COM file using the optional file name <filename>.COM on the optional drive d:. The file name of the first module loaded is used if the optional file name is not included. The drive always defaults to the currently logged-on disk for execution unless the drive name d: is included. The file type .COM may be specified but is the default (and only valid) file type.

9.2.3 T

This prints the global symbol table after all modules on the command line list have been loaded, as seen here.

```
*T

SYMBOL TABLE (UNDEF=****)

ATTN   01E3    DRQ    0215    HOMEDK 0103    KKPLC  02A0
MINUS  0276    PLUS   0204    STQY   01EA    SUBQ   013B
TTYTRU 02BB    ZZZZ   0118
```

If the E command is used on the options list, and one or more global symbols are unresolved after linking and loading all the modules in the file name list, then EXLINK will display an error message and return to interactive mode. However, if the E command is input interactively, and one or more global symbols are unresolved, then EXLINK displays the same error message as a warning, and writes the COM file anyway with the unresolved symbols.

9.3 Other features of EXLINK

9.3.1 Working memory or the "loading zone" is filled with zeros before loading begins. Hence, a COM file written from modules starting at an address greater than 100 hex, while not recommended, will execute properly after being called from CP/M. Execution starts at the module with the lowest starting address. The preceding zeros are decoded as NOP (no op) instructions by the CPU, and cause the system to "fall through" to the first module.

9.3.2 EXLINK does not permit modules to be loaded in RAM between 0 and 100 hex (this would wipe out the CP/M work area) nor in RAM occupied by EXLINK itself. A warning is issued if an attempt is made to load modules at addresses above EXLINK's highest address. The user may override this warning. However please note that by using this override, it is possible to overwrite the CP/M BDOS and destroy EXLINK's disk access capability. So, extreme care should be taken when loading modules in RAM above EXLINK.

9.3.3 In the running of EXLINK, certain messages are displayed. BEG ADDR and END ADDR specify the absolute location or boundary limits of a module in RAM. UNDEF. SYM refers to the number of symbols not yet resolved.

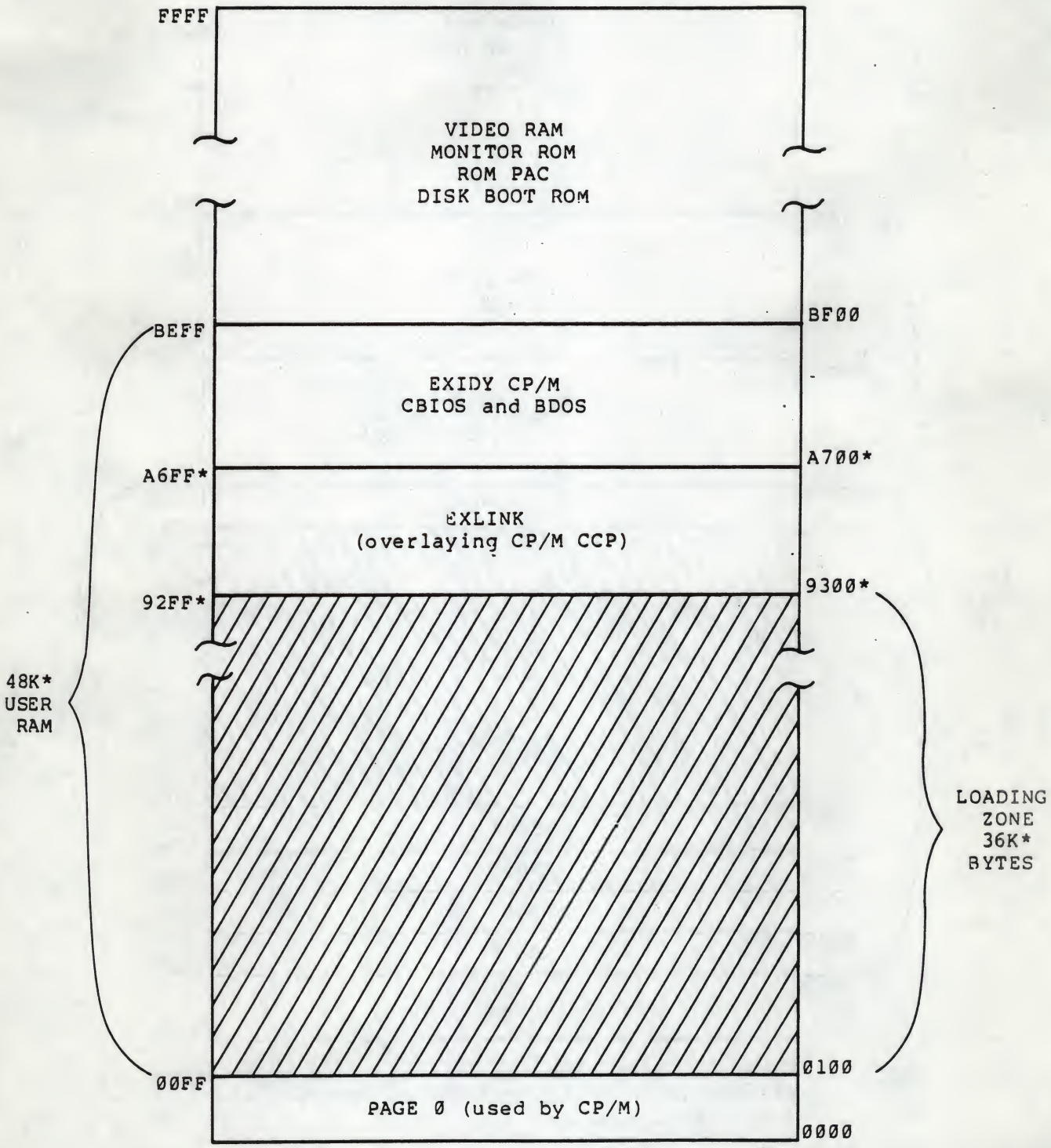
10 SAMPLE RUNS

10.1 Batch mode linking example

Suppose the EXASM assembler had assembled four modules of source code with global references between them. The four object modules are accessible as CP/M .OBJ files on disk and are ready for linking. All are ORGed at 0 but the starting address is 100. These modules are named MAIN.OBJ, SUB1.OBJ, SUB2.OBJ, SUB3.OBJ and all are on drive A except SUB2.OBJ which is on drive B. (Refer to the Memory Map diagrams). Call up EXLINK this way (user input is underlined, and carriage returns are understood at the end of each line):

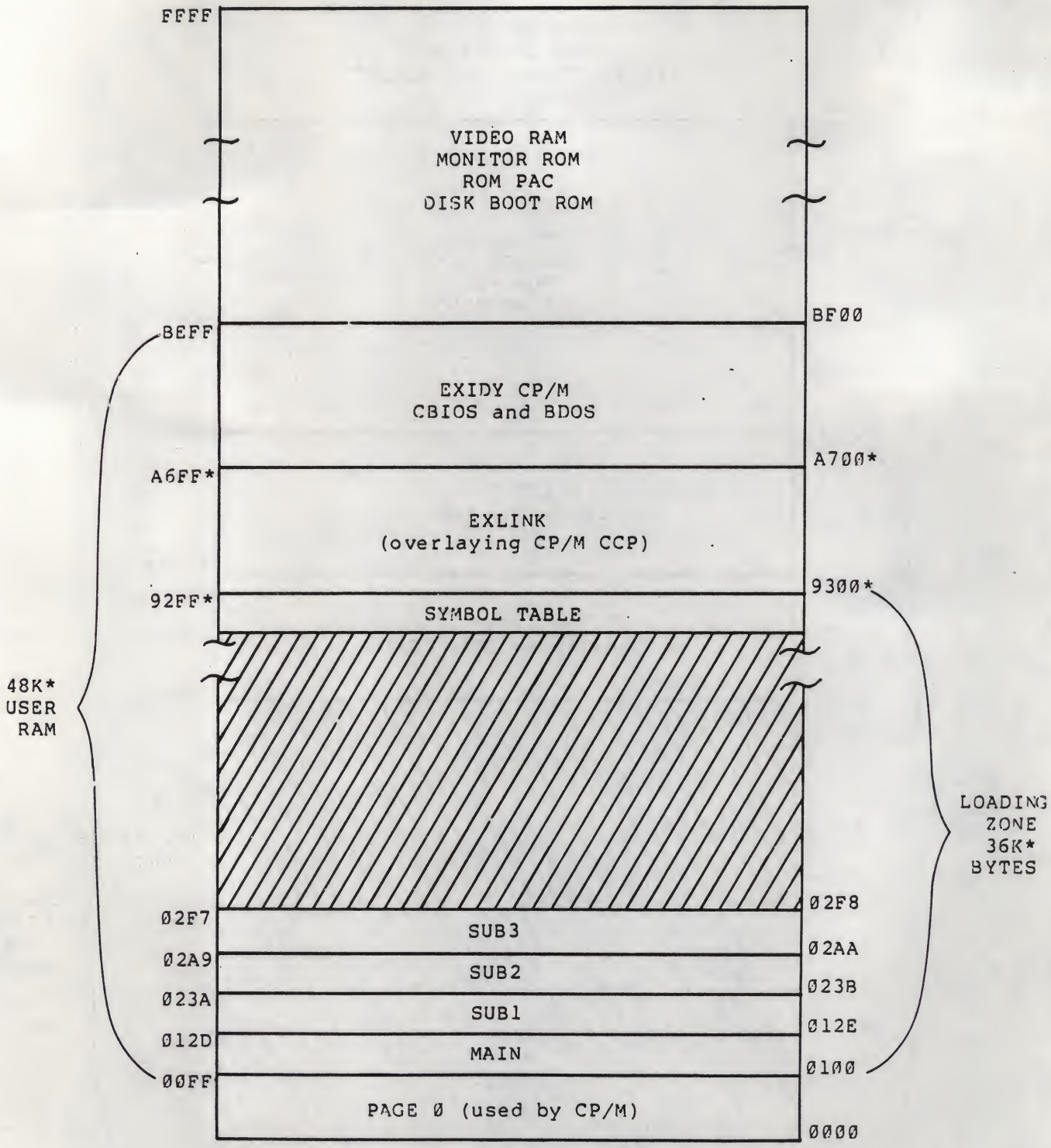
```
A>EXLINK MAIN,SUB1,B:SUB2,SUB3/A=100,E=NEWNAME,T

Exidy Relocating Linking Loader.
Copyright (c) 1980 Exidy Inc. ver 2.1

Starting offset is 100

*L MAIN

BEG ADDR 0100
END ADDR 012D
UNDEF SYM 04

*L SUB1

BEG ADDR 012E
END ADDR 023A
UNDEF SYM 03

*L B:SUB2

BEG ADDR 023B
END ADDR 02A9
UNDEF SYM 06

*L SUB3

BEG ADDR 02AA
END ADDR 02F7
UNDEF SYM 00

*T

SYMBOL TABLE (UNDEF=****)

ATTN   01E3    DRQ    0215  HOMEDK  0103     KKPLC  02A0
MINUS  0276    PLUS   024   STQY    01EA     SUBQ   013B
TTYTRU 02BB    ZZZZ   0118

*E NEWNAME
NEWNAME.COM SAVED, RECS WRITTEN=04
A>
```

MEMORY MAP
48K SORCERER
BEFORE MODULES ARE LOADED



Note: All addresses are in hexidecimal and refer to a 48K CP/M. This illustration is not to scale.
*-Approximate value

MEMORY MAP
48K SORCERER
AFTER MODULES ARE LOADED



Note: All addresses are in hexidecimal and refer to a 48K CP/M. This illustration is not to scale.
*-Approximate value

The console I/O shown above is produced automatically after the CP/M command line is typed in by the user. This is an example of batch mode linking. Because all modules are ORGed at 0, the only offset involved is the starting one (100 Hex), specified with the A= option. The symbol table is placed in the default RAM area. MAIN is loaded from 100H (ORG 0+100H) to 12D. It has four unresolved external references. SUB1 is loaded immediately after MAIN, at 12E (the sum of ORG 0 + the last byte 12D+1 + the offset 0, not specified). This process continues for each module loaded. The T option displays the global symbols and each of their addresses. The E option writes to disk all RAM from 100H to the highest address, unless prevented by undefined symbols.

Since the E command is included with the name NEWNAME in the options list, EXLINK writes a .COM file named NEWNAME.COM. This file is the memory image formed from loading the modules MAIN.OBJ, SUB1.OBJ, SUB2.OBJ, and SUB3.OBJ. In other words, the output of EXLINK is a CP/M file (in this case named NEWNAME.COM) corresponding to the memory image from 100H to the highest address loaded (2F7) after EXLINK converts the four object modules into absolute machine executable code with global references resolved.

10.2 Interactive mode linking example

The following sample run creates the same .COM file interactively but this time it is called MAIN.COM.:

```
A>EXLINK

Exlink Relocating Linking Loader.
Copyright (c) 1980 Exidy Inc. ver 2.1

Starting offset is 0          (default offset is 0).

*L MAIN 100                   (the user's response loads
                               MAIN.OBJ with starting
                              offset of 100.)

BEG ADDR 0100
END ADDR 012D
UNDEF SYM 04

*L SUB1

BEG ADDR 012E
END ADDR 023A
UNDEF SYM 03

*L B:SUB2

BEG ADDR 023B
END ADDR 02A9
UNDEF SYM 06
```

```
*L SUB3

BEG ADDR 02AA
END ADDR 02F7
UNDEF SYM 00

*T

SYMBOL TABLE (UNDEF=****)

ATTN   01E3    DRQ    0215    HOMEDK 0103    KKPLC  02A0
MINUS  0276    PLUS   0204    STQY   01EA    SUBQ   013B
TTYTRU 02BB    ZZZZ   0118

*E

MAIN.COM SAVED, RECS WRITTEN=04

A>
```

(Now a file is written named MAIN.COM since the optional file name isn't included.)

The above console I/O is much like the first example. The difference is that after each * prompt, the user enters each command interactively.

11 ERROR MESSAGES

Thirteen error conditions cause an error message to be displayed. Some errors are fatal, and control returns to CP/M. Others are non-fatal and return to interactive mode input (with the * prompt).

**** CHKSUM ERROR ****

Checksum error. The Checksum computed from an input file record doesn't agree with the one originally recorded. This error is rare since errors of this type are usually caught by CP/M's disk I/O first.

**** DBL DEF ERROR ****
SYM: symbolname

Double definition of a global symbol. This error occurs when a symbol declared global and defined in one module is declared and defined in another. The particular symbol is shown following "SYM:". Of the two definitions, the first one is used as the symbol's value.

**** SYM TAB OVERWRITE ERROR ****

Attempt to overwrite the loader symbol table. A module attempts to load over the global symbol table and is prevented by this fatal error. CP/M warm boots at this point.

**** PROTECT RAM LOAD ERROR ****

Attempt to load outside "safe" RAM area. Protected RAM is page 0 (0000-00FF hex), and the RAM occupied by EXLINK. This error is also fatal and warm-boots CP/M.

**** SYM TAB OVFLO ERROR ****

Symbol table overflow. Table reaches 100H. This is a fatal error causing CP/M to warm-boot.

**** SYNTAX ERROR ****

This message is displayed if a command other than L, T, Q, or E is entered in interactive mode or if non-hexidecimal digits are used when hex is expected. It is also displayed if EXLINK cannot make sense of the CP/M command line in batch mode.

**** DISK WRITE ERROR ****

This error occurs when writing the .COM file if either the diskette directory or file space is full, or any other write fault occurs. This error is fatal, causing CP/M to warm start.

**** BAD FILE TYPE ERROR ***

If the user requests EXLINK to load a file with a filetype other than .OBJ, this error occurs. This error also occurs when output files (associated with the E command) are not of type .COM. The error is non-fatal.

**** UNDEF SYM ERROR ****
**** UNDEF SYM WARNING ****

If the E command (exit with .COM file write) is used when undefined symbols are still outstanding, this message appears. In batch mode, this error causes a changeover to interactive mode in which case the prompt (*) is displayed and further user input is expected. In interactive mode, the message is displayed as a warning but the .COM file is written anyway.

**** BEG ADDR NOT 100H ERROR ****
**** BEG ADDR NOT 100H WARNING ****

This message is displayed when the E command is used and the modules loaded don't start at 100 hex (the start of CP/M's transient program area, TPA). In batch mode it is a non-fatal error causing it to change to interactive mode. In interactive mode it is merely a warning and allows the .COM file to be written anyway for starting addresses greater than 100H.

**** LOAD ABOVE EXLINK ERROR ****
ADDR: xxxx DO IT ANYWAY (Y/N)?

This message expects user input to enable/disable module loading above EXLINK program. If N is entered, the link is aborted. If Y is entered, EXLINK proceeds to load the module and doesn't check again for modules loaded above EXLINK. The address indicated is the first memory location encountered above EXLINK.

**** .OBJ FILE NOT FOUND ERROR ****
FILE: filename

If an input file name is given to EXLINK and cannot be found on the drive indicated, this message appears. The file name in question is displayed on the following line.

**** NOTHING TO SAVE ERROR ****

If the E command is given before any modules have been loaded, this message appears.

12 EXAMPLE OF THE COMPLETE EXASM AND EXLINK

Here is the assembly by EXASM of two program segments, MODUL1 and MODUL2. Note the unresolved external global references in the object code of MODUL1 (indicated by asterisks). Notice the trailing apostrophes in the object code of MODUL2, referring to relocatable addresses.

Next, with EXLINK we link the two modules and load them under the name TEST.COM. After the linking of MODUL1, EXLINK tells us there is one undefined symbol (the external global XXX). After the linking of MODUL2, we see that the reference has been resolved (because EXLINK reports no undefined symbols). Notice that MODUL2 is assembled with starting address of 0000.

To compare code, we use the CP/M DUMP program. Notice that the unresolved CALLs from MODUL1 are now calls to address 010CH. (CD0C01). CD is the hex code for the CALL instruction. The CPU knows that the next two bytes will be an address, with the low-order or least significant byte first. That is, CD0C01 means CALL 010CH. (At location 010CH is the routine that we named XXX.)

Notice that EXLINK starts MODUL2 at 010CH, directly after the last address of the object code of MODUL1. (The instruction C3 is at address 0109H, while the jump address 0000 is at 010A and 010BH.)

After the command file TEST.COM has been saved on disk, it can be executed merely by typing its name on the command line. What the program does is use CP/M's BDOS to print the message three times to the console.

```
A>EXASM MODUL1/LE

EXIDY Z80 Assembler -  version 2.1
Copyright (C) 1980 by EXIDY INC
PASS 2
MODUL1                                          EXIDY Z80 ASSEMBLER V 2.1  PAGE
  ADDR  OBJECT    ST # SOURCE STATEMENT

                  0001          NAME    MODUL1
                  0002          GLOBAL  XXX       ;EXTERNAL
                  0003          ORG     100H      ;ORG AT CPM TPA
 '0100  CDFFFF*   0004          CALL    XXX       ;PRINT MESSAGE
 '0103  CD0101*   0005          CALL    XXX       ;PRINT MESSAGE
 '0106  CD0401*   0006          CALL    XXX       ;PRINT MESSAGE
 '0109  C30000    0007          JP      0         ;WARM-START CPM


ERRORS=0000

WARNINGS=0000

A>EXASM MODUL2/LE

EXIDY Z80 Assembler -  version 2.1
Copyright (C) 1980 by EXIDY INC
PASS 2
MODUL2                                          EXIDY Z80 ASSEMBLER V 2.1  PAGE
  ADDR  OBJECT    ST # SOURCE STATEMENT

                  0001          NAME    MODUL2
                  0002          GLOBAL  XXX       ;INTERNAL
 '>0000           0003 XXX:
 '0000  211500'   0004          LD      HL,MSG    ;POINT HL REGISTER TO
                  0005                            ;MESSAGE TEXT
 '>0003           0006 LOOP:
 '0003  7E        0007          LD      A,(HL)    ;GET A CHARACTER
 '0004  B7        0008          OR      A         ;DONE?
 '0005  CA1400'   0009          JP      Z,DONE    ;YES, EXIT
 '0008  5F        0010          LD      E,A       ;NO, PUT CHARACTER IN BDOS
                  0011                            ;(REGISTER E)
 '0009  0E02      0012          LD      C,2       ;GET WRITE CONSOLE CHARACTER
                  0013                            ;FUNCTION CODE
 '000B  E5        0014          PUSH    HL        ;SAVE HL
 '000C  CD0500    0015          CALL    5         ;CALL CPM BDOS TO WRITE CHAR
                  0016                            ;TO CONSOLE
 '000F  E1        0017          POP     HL        ;BRING HL BACK
 '0010  23        0018          INC     HL        ;NEXT CHARACTER IN MESSAGE
 '0011  C30300'   0019          JP      LOOP      ;CONTINUE FOR ALL BYTES
 '>0014           0020 DONE:
 '0014  C9        0021          RET
 '0015  0D0A      0022 MSG:     DEFB    0DH,0AH ;CARRIAGE RETURN/LINE FEED
 '0017  54455354  0023          DEFM    'TESTING'
        494E47
 '001E  00        0024          DEFB    0
                  0025          END


ERRORS=0000

WARNINGS=0000
```

```
A>EXLINK MODUL1,MODUL2/A=0,E TEST,T

EXIDY RELOCATING LINKING LOADER.
 OPYRIGHT (C) 1980 EXIDY INC. VER 2.1
STARTING OFFSET IS 0

*L MODUL1

BEG ADDR 0100
END ADDR 010B
UNDEF SYM 01

*L MODUL2

BEG ADDR 010C
END ADDR 012A
UNDEF SYM 00

*T
SYMBOL TABLE (UNDEF=****)

XXX     010C

*ETEST

TEST.COM SAVED, RECS WRITTEN= 01

A>DUMP TEST.COM


0000 CD 0C 01 CD 0C 01 CD 0C 01 C3 00 00 21 21 01 7E
0010 B7 CA 20 01 5F 0E 02 E5 CD 05 00 E1 23 C3 0F 01
0020 C9 0D 0A 54 45 53 54 49 4E 47 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

A>TEST

TESTING
TESTING
TESTING
A>
```

PART III: DEVCNVRT

8. INTRODUCTION

DEVCNVRT is a CP/M compatible utility program that converts cassette files created by the Exidy Development-PAC into disk files accessible to CP/M development systems. When the DEVCNVRT program is invoked from CP/M, a filename and filetype given on the command line become the name and type of the newly created disk file. Then, a cassette containing a Development-PAC file is read into the Sorcerer either under motor control, or by manual control of tape pauses. The pauses are necessary to allow time for each file block read from cassette to be written to disk, without passing over blocks on tape. After all blocks have been read in, the program inserts a CP/M end of file character (cntl-Z) where the Development-PAC's end of file character (cntl-C) was found, and closes the disk file. The new disk file can be manipulated by CP/M assemblers and linkers such as EXASM and EXLINK. Cassette files containing DEVELOPMENT PAC assembly source (from the Pac text editor ED) or object code (from the PAC assembler ASM) can be converted.

14 CONVERTING A FILE FROM CASSETTE TO DISK

To convert a Development-PAC cassette file to a CP/M compatible disk file, follow these steps. For clarification, user input is underlined.

After the CP/M A> prompt, enter DEVCNVRT in the command line, following it with a space. Then type in the filename and filetype, separating them with a period. This becomes the name and filetype of the newly created disk file, as shown here:

A> DEVCNVRT FILENAME.ASM <Return>

If the filename and type are omitted, or if the filetype is not .ASM .OBJ or HEX, an error message is displayed. Only assemby (.ASM) or object (.OBJ, HEX) cassette files are supported.

After an acceptable filename and type are entered, the program asks the user if he has cassette motor control. The user responds by typing Y (for yes) or N (for no), as seen here:

DO YOU HAVE CASSETTE MOTOR CONTROL (Y/N)? Y

(Motor control requires the use of the Serial Cassette Data Cable, DP 4005).

Then, a message is displayed, telling the user to rewind the cassette. If the user has motor control, set the recorder to play, hit any key and the file is automatically converted without manual control of the cassette recorder motor.

REWIND CASSETTE. WHEN READY TO PLAY TAPE, HIT ANY KEY.<any key>

If the user does not have motor control, having only manual control, the following messages are displayed:

MANUAL MOTOR CONTROL MUST BE USED.

STOP TAPE RECORDER (OR PAUSE) IMMEDIATELY WHEN THE "STOP TAPE!!!" MESSAGE IS DISPLAYED. TAPE MAY BE TURNED ON AT LEISURE WHEN "PLAY TAPE." MESSAGE APPEARS.

REWIND CASSETTE. WHEN READY TO PLAY TAPE, HIT ANY KEY.<any key>

Rewind the cassette tape to the beginning of the file, and the message "PLAY TAPE" is displayed. Start the recorder after this message is displayed. When "STOP TAPE!!!" is displayed, THE CASSETTE PLAYER MUST BE STOPPED OR PAUSED IMMEDIATELY UNTIL THE NEXT "PLAY TAPE" PROMPT APPEARS. A few moments after the cassette player is stopped, the "PLAY TAPE" message should be displayed on the screen. The user does not need to respond immediately and may press the PLAY button to start the tape at his convenience. The start/stop procedure is repeated for each disk/tape block until the program is completed.

After the conversion is successfully completed, the console displays the following message and returns to CP/M.

SUCCESSFUL DISK WRITE

15 EXAMPLE RUN

The following are examples of console I/O when using the DEVCNVRT conversion program. Underlined information is entered by the user.

```
A>DEVCNVRT SAMPLE.ASM<Return>

EXIDY CASSETTE TO DISK FILE TRANSFER PROGRAM
FOR CASSETTES CREATED BY THE DEVELOPMENT-PAC.
VER. 1.0

DO YOU HAVE CASSETTE MOTOR CONTROL (Y/N)?Y

REWIND CASSETTE. WHEN READY TO PLAY TAPE,
HIT ANY KEY.<any key>

SUCCESSFUL DISK WRITE
```

```
A>DEVCNVRT SAMPLE.ASM <Return>

EXIDY CASSETTE TO DISK FILE TRANSFER PROGRAM
FOR CASSETTES CREATED BY THE DEVELOPMENT-PAC.
VER. 1.0


DO YOU HAVE CASSETTE MOTOR CONTROL (Y/N)?N

MANUAL MOTOR CONTROL MUST BE USED.

STOP TAPE RECORDER (OR PAUSE) IMMEDIATELY
WHEN THE "STOP TAPE!!!" MESSAGE IS DISPLAYED.
TAPE MAY BE TURNED ON AT LEISURE WHEN
"PLAY TAPE." MESSAGE APPEARS.

REWIND CASSETTE.   WHEN READY TO PLAY TAPE,
HIT ANY KEY.<any key>

PLAY TAPE.
(or)
*** STOP TAPE!!!! ***

SUCCESSFUL DISK WRITE

A>
```

16 ERROR MESSAGES

If an error occurs, one of the following messages is displayed describing the error.

```
FILE NAME NOT GIVEN, OR FILE TYPE
NOT "ASM", "HEX", OR "OBJ".
```

This error occurs if the DEVCNVRT command line does not contain a filename and filetype after DEVCNVRT, or if the filetype is not ASM, HEX, or OBJ.

```
DISK IS FULL, WRITE INCOMPLETE.
```

This error occurs if the CP/M diskette does not have enough room for the new file. The incomplete disk file is closed, and control returns to CP/M.

```
TAPE CRC ERROR
```

This error results if 8-bit CRC generated during the cassette write operation does not agree with the CRC generated during the read back operation. This error is usually the result of an improper tone or volume setting on the tape recorder. Try different volume and tone settings on your tape recorder.

APPENDIX A: EXASM Abstract Reference

A.1 EXASM call format

EXASM <sourcefile>[,<objectfile>][,<printfile>] [/<options>]

A.2 Options

C - Generate cross-reference.
D - Listing to disk.
E - "Ecology" or compressed listing
F - Set form-feed option.
G - Suppress generated text.
K - No listing or cross-reference.
L - Listing to list device.
N - No object output.
O - Object output.
S - No form feeds.
T - List to console.
W - Don't print warnings.

A.3 Pseudo-op syntax

A.3.1 Data Generation

DEFB/DEFM/DB : <label> DEFB n[,n,n...]

<label> DEFW <expr>

<label> DEFS <expr>

A.3.2 Source control

<label> IF nn

ENDIF

INCLUDE <filename[.<type>]>

Object control

<label> PSECT <opr>

<label> ORG <expr>

<label> END

<label> NAME <string>

A.3.3 Listing control

EJECT

TITLE

PAGE x

LIST

NLIST

A.3.3.1 LIST and NLIST with operands: options:

G - Don't print text.

W - Don't print warnings.

E - "Ecology" (suppression of form feeds and ejects).

Examples:

```
LIST GW
NLIST GW
```

A.3.4 Symbol control

<label> EQU <expr>

<label> DEFL <expr>

A.3.5 Linking Control

<label> GLOBAL <symbol>

These three may be used in place of the GLOBAL pseudo-op to specify an external global symbol:

EXTERNAL, EXTERN, EXT

These four may be used in place of the GLOBAL pseudo-op to specify an internal global symbol:

INTERNAL, INTERN, INT, PUBLIC

You may use any of the forms interchangeably.

APPENDIX B: EXLINK Abstract Reference

A.1 EXLINK call format

EXLINK [<file1>][,<file2>][,<file3>]...[/options]

B.2 Offset for loading first module is determined by sum of:

A=XXXX	where XXXX is the starting offset from call option list
ORG YYYY	(given in source module)
ZZZZ	(offset number) given in load command

B.3 Offset for loading subsequent modules is determined by:

ORG YYYY	(given in source module)
ZZZZ	(offset address on command line)
Default:	End address of previous module + 1.

B.4 Interactive Mode Options

L [d:]<filename> [ZZZZ]	Relocates file at optional starting offset ZZZZ.
T	Prints current global symbol table.
E	Writes .COM file to drive d:.
Q	Exits without writing .COM file.

B.5 Batch Mode Options

A=XXXX	
E	Exits loader and writes a .COM file to disk.
T	Prints global symbol table, after loading modules

APPENDIX C: DEVCNVRT Abstract Reference

C.1 DEVCNVRT call format

DEVCNVRT FILENAME.ASM